Aeroelastic flutter as a multiparameter eigenvalue problem

by

Arion Pons

Submitted in partial fulfilment of the requirements
for the degree of

Master of Engineering

in the Department of Mechanical Engineering
at the University of Canterbury.

2015

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I am grateful to the University of Canterbury for a CWF Hamilton Master's Scholarship, to my parents for their support, and to Stefanie Gutschmidt my supervisor. Thanks are also due to A. Muhič and B. Plestenjak of the University of Ljubjana for the use of their *MultiParEig* code before its publication.

ABSTRACT

In this thesis we explore the relationship between aeroelastic flutter and multiparameter spectral theory. We first introduce the basic concept of the relationship between these two fields in abstract terms. Then we expand on this initial concept, using it to devise visualisation methods and a wide variety of solvers for flutter problems. We assess these solvers, applying them to real-life aeroelastic systems and measuring their performance. We then discuss and devise methods for improving these solvers. All our conclusions are supported by a variety of evidence from numerical experiments. Finally, we assess all of our methods, providing recommendations as to their use and future development.

We do achieve several things in this thesis which have not been achieved before. Firstly, we solved a non-trivial flutter problem with a direct solver. The only direct solvers that have previously been presented are those that arise from classical flutter analysis, which applies only to very simple systems. Secondly, and as an extension of this first point, we solved a system with Theodorsen aerodynamics (approximated by a highly accurately) with a direct solver. This was achieved in an industrially competitive time (0.2s). This has never before been achieved. Thirdly, we solved an unstructured multiparameter eigenvalue problem. Unstructured problems have not been considered before, even in theoretical literature. This result is thus of significance both for multiparameter spectral theory and aeroelasticity. However, the single most important contribution of this thesis is the opening of a whole new field of study which stretches beyond aeroelasticity and into other industries: the treatment of instability problems using multiparameter methods. This field of research is wide and untrodden, and has the potential to change the way we analyse instability across many industries.

NOMENCLATURE

$\imath$	imaginary unit		
$\mathrm{Re}(c)$	real part of c	$\mathrm{Im}(c)$	imaginary part of c
c^T	matrix transpose of c	c^H	Hermitian transpose of c
$\dot{c}$	time derivative of c	$\partial_x c$	derivative of c with respect to x
$\otimes$	Kronecker product	$c\|_Q$	evaluation of c at condition Q
$[a; b]$	vertical concatenation of a and b	$[a, b]$	horizontal concatenation of a and b

Matrices will be denoted in regular font, vectors in boldface, and scalars in italics.

PREFACE

It is important that you read this preface. This thesis presents a synthesis of two areas of study which have not had any previous contact, and it is unlikely that the reader will have expertise in both of these areas. These areas are aeroelasticity and multiparameter spectral theory. This thesis has been written with aeroelasticians (i.e. engineers) in mind, and it concerned primarily with applying the tools available in multiparameter spectral theory to aeroelastic problems – as opposed to making any theoretical developments in multiparameter spectral theory *per se*. We will, in fact, often advance beyond what has currently been done in the theoretical literature on this subject (especially as regards our algorithms for nonlinear multiparameter eigenvalue problems). However, we would like to note up-front that readers with expertise in abstract mathematics will be unsatisfied by the standard of proof employed in this work. We will not attempt to prove results (e.g. convergence of iterative methods) in any other way than the application to a practical problem.

The structure of this thesis is relatively simple. Chapter 1 introduces the basic relationship between aeroelasticity and multiparameter spectral theory, upon which we will build everything else in the thesis. Chapter 2 introduces the physical aeroelastic systems that we will be working with. Chapters 3-6 are then go over a variety of difference system classes, and their respective solution methods. Each of these chapters is relatively distinct and may be read without much reference to the others. For this reason we have included a bibliography at the end of each chapter, in the place of one large bibliography at the very end. Chapter 7 presents a final conclusion and assessment of the previous six chapters, as well as a discussion on some further interesting phenomena and unsolved problems. Aeroelasticians or other engineers wanting to implement the methods presented in this thesis may find it easiest to locate the system of interested in the systems index (Chapter 2), and then

Mathematicians may be particularly interested in Chapters 1 and 6, where our most novel theoretical work occurs. They may also be interested in Chapter 2 as a source of motivation for their own research in multiparameter spectral theory. Previous to this thesis there has been no clear motivation for the analysis of high-dimension, large polynomial, semis-

structured or unstructured multiparameter eigenvalue problems. All readers, theoretical or practical, will be interested in Chapter 7, where we summarise our developments and look forward to future avenues of application and research.

Chapter 1
Initial manoeuvres

1.1 INTRODUCTION AND SCOPE

Aeroelasticity is the field of study concerned with the interaction of an elastic structure in an airstream. As such, aeroelasticity can be seen as a subfield of the more general study of fluid-structure interaction (FSI), being concerned largely with the case when the working fluid is air. Alternately but equivalently, one might also see the field of aeroelasticity as the synthesis of the two fields of aerodynamics and structural dynamics – and this is indeed the historical origin of the field [1]. Aeroelasticity is thus of great relevance to the aeronautics industry: Hodges [2] notes that "the solution of many aeroelastic problems is a basic requirement for achieving an operationally reliable and structurally optimal [aerospace] system". However, its applications in other areas – in particular, turbomachines and wind turbines – should not be overlooked.

One of the prime concerns in modern aeroelasticity is how to predict and control aeroelastic instability. An instability, in this context, may be defined as an event during which the structure in question becomes self-exiting. When this occurs dynamically (as an oscillatory instability) it is termed *flutter*, and when it occurs statically (as a non-oscillatory instability) it is termed *divergence* [3]. Divergence may be seen as a subsidiary form of flutter, and in this work we will use the term *flutter* to denote either instability. The term *dynamic flutter* will be used to denote flutter that is specifically oscillatory. In aeronautical systems, a number of different classes of flutter events may be observed. Some of the more important classes are [2–5]:

Classical flutter – a lifting surface at normal angle-of attack is exposed to subsonic flow.

Stall flutter – a lifting surface at high (stalling) angle-of-attack is exposed to subsonic flow.

Supersonic flutter – a lifting surface is exposed to supersonic flow.

Control surface flutter – an actuated control surface is exposed to flow.

Panel flutter – a non-lifting surface is exposed to flow.

Of course, for each of these classes there exist a large number of different mathematical models, for modelling different situations with different levels of accuracy. One easy distinction to make is between models based on linear and nonlinear systems[1]. In a linear aeroelastic system, the onset of flutter or divergence can be formulated in the well-known stability criterion:

$$\mathrm{Im}(\chi) > 0 \text{ for stability} \qquad (1.1.1)$$

where χ are the time-eigenvalues of the system according to the Fourier transform $h(t) = \bar{h}\exp(\iota\chi t)$ for the system coordinate h. These eigenvalues are often nondimensionalised with respect to airspeed, but this is not relevant at this stage. In graphical terms, Eq. 1.1.1 means that eigenvalues in the upper half-plane of an Argand diagram are stable, and those in the lower half-plane are unstable[2]. Flutter occurs when the system parameters (airspeed, air density, etc.) are such that the system is on the point of crossing from stability into instability or vice-versa: that is, where $\mathrm{Im}(\chi) = 0$. A given system may have many flutter points: each point is defined by a modal frequency and an airspeed value, with the air density and other parameters typically being fixed. Flutter points are always ordered by increasing airspeed: the first flutter point is the flutter point that occurs at the lowest (positive) airspeed value, etc. Dynamic flutter and divergence points are ordered separately, and those that occur at negative airspeed or frequency are irrelevant. Typically only the first flutter point and first divergence point are of industrial relevance. In an aircraft, the first aeroelastic instability alone will define the flight envelope of the aircraft – both flutter and divergence will typically result in catastrophic failure and the destruction of the flight vehicle [2].

In nonlinear systems, aeroelastic instabilities can be described by a number of nonlinear phenomena, including Hopf bifurcations and limit cycle oscillations [6,7]. The range of nonlinear models used in aeroelasticity is vast – ranging from analytical section models with

[1] Please note the difference between a *nonlinear eigenvalue problem* and a *nonlinear system*.

[2] Some readers may prefer to rotate their eigenvalues 90° in the Argand plane (i.e. $\lambda = \iota\chi$) in which case the right-half plane is the unstable half-plane. However, we adopt the convention of Eq. 1.1.1 throughout this work.

simple nonlinearities, to fully-coupled FEA/CFD simulations. However, one should not suppose that the introduction of nonlinear models in aeroelasticity has made linear models obsolete, in industry or research: significant effort is still going into devising better methods for linear flutter point prediction. With the growing emphasis on optimisation of aircraft structures [8] and active aeroelastic control [9], low computation time for aeroelastic problems is imperative. Many computational models for flutter are based on linear theories – even for such turbulent systems as bridge decks under wind loading [10–13]. Most industrial aeronautical flutter solvers, such as MSC Nastran[3] or ZAERO[4], use linear potential-flow aerodynamics and solve their aeroelastic systems via frequency domain solvers [14,15]. Faster algorithms for such systems would have benefit to a wide range of industry. In recent years there has also been a proliferation of analytical models for novel aerodynamic / aeroelastic systems. These include models for camber-flutter in conformable wings [16,17], more accurate analytical flow models for helicopter blades [18–20], and a variety of biomechanical aeroelastic models involving fish [21], birds [22] and insects [23].

Our work will be concerned with linear aeroelastic systems, but even in this context it should be noted that Eq. 1.1 is not the only stability criterion that can be used to characterise aeroelastic flutter. Different ones can be devised, and even for a given criterion there may be a number of ways of devising and solving the associated system. This leads us into the study of aeroelastic *methods*, and it is worthwhile spending some time elaborating on these. There are four established methods in modern aeroelastic analysis: the *p-method*, *classical flutter analysis*, the *k-method* (or *U-g method*, or *V-g method*) and the *p-k method* [2]. The p-method is the most fundamental, and its use traces back to the earliest flutter analyses performed by British aerodynamicists after the First World War [24,25]. For this reason it is often referred to as the 'British method' in older literature [26]. The p-method simply involves solving the aeroelastic system for χ (or usually an equivalent nondimensional parameter p) over a range of airspeed values, and interpolating or otherwise estimating the points where Eq. 1.1 holds. If used with accurate aerodynamic and structural models, it can predict both the flutter points of the aeroelastic system and its modal characteristic at any airspeed. Its main disadvantage is that is requires an

[3] 'NASTRAN' is a registered trademark of the National Aeronautics and Space Administration.

[4] 'ZAERO' is a registered trademark of ZONA Technology Inc.

aerodynamic model that is valid for the case when $\mathrm{Im}(\chi) = 0$, and this is not always available [2]. This spurred the development of classical flutter analysis. This method involves deriving aeroelastic models based on the presumption that the system is at its flutter points ($\mathrm{Im}(\chi) = 0$), and then solving these models (usually by a fixed-point iteration) for the conditions at which this is in fact true. One only has to have an aerodynamic model that is valid at the flutter point; such models are much easier to derive or obtain experimentally. Classical flutter analysis produces the same estimates of the flutter point locations as the p-method (for the same aeroelastic model, and assuming perfect convergence of the iterative method). However, it gives no information on the behaviour of the system away from its flutter points.

In parallel with the development of classical flutter analysis was the development of the k-method. In modern literature this is also referred to as the U-g method or V-g method, and in older literature as the 'American method'. This is due to its development in America and subsequent contrast with the 'British method' [14]. The k-method introduces a fictitious structural damping term into the aeroelastic system, and then (over a range of airspeed values) solves for the fictitious structural damping that would be required to bring the system to flutter at the given airspeed value. This produces modal-damping paths that are similar to those of the p-method, but the instability criterion is now different. Flutter is defined as occurring where the fictitious structural damping becomes zero [2]. Like classical flutter analysis, the k-method only requires an aerodynamic model that is valid at the flutter point. If the aeroelastic models are the same, then the k-method should produce the same estimates of the flutter point locations as the p-method and classical flutter analysis. However, in a landmark paper Hassig [27] showed that the k-method can produce seriously wrong results at airspeeds away from the flutter point, and can also mispredict which mode will flutter. As a solution to this, Hassig devised the p-k method. This method is in some sense a synthesis of the p- and k- methods: it involves performing a p-method analysis on a system where the aerodynamic components of the system are defined only for $\mathrm{Im}(\chi) = 0$, but the structural components are defined for all χ. The p-k method still only has to have an aerodynamic model that is valid at the flutter point, but gives much better subcritical and supercritical predictions than the k-method.

As can be seen, the difference between these methods is partly a matter of how the solution is solved mathematically, and how the real physical system is transferred into a mathematical model in the first place. In recent years there has been a proliferation of new aeroelastic methods. There has been an interest in the application of concepts from robust control theory; in particular, the linear fractional transformation and the structured singular value (μ). This yielded a series of methods based on dynamic pressure perturbation, including the μ-method by Lind and Brenner [28,29], the μ-k method by Borglund [30–33], the μ-ω method by Gu et al. [34,35] and most recently, the μ-p method again by Borglund [36]. The prime advantage of these μ-type methods is that they facilitate the propagation of detailed uncertainty distributions through the aeroelastic system. This allows a worst-case flutter speed estimate to be made in a system with high uncertainty [36]. However, some of these methods can fail to predict flutter speeds accurately due to problems with dynamic pressure perturbations involved in the solution process [35]. Other developments have come from other fields. Afolabi [37,38] characterised coupled-mode flutter as a loss of eigenvector orthogonality, using methods from catastrophe theory. Gu et al. [39] applied a genetic algorithm to an existing μ-method, and a number of authors [40–43] have applied neural networks to the detection of flutter points. Haddadpour and Firouz-Abadi [44] developed the pp-method, an extension of the p-method for functions with complicated dependence on the dimensionless eigenvalue p. Namini et al. [11] developed the pk-F method, a variant of the p-k method specifically for analysing finite-element models of bridges. Chen [45] modified the p-k method to produce the g-method. Irani and Sazesh [46] used stochastic analysis to devise a method of identifying flutter points based on the evaluating the system's response variance over a given airspeed range.

It is in the context of these developments that we propose our method of analysing flutter problems. The central methodological contribution of this thesis is the concept that the solution of an aeroelastic system for its flutter points is nothing other than a multiparameter eigenvalue problem. We will show the simple link between the aeroelastic stability problem and that discipline of abstract mathematics known as multiparameter spectral theory. This has not been done before. In this sense the analysis that we will perform is entirely different from any of the methods that have been proposed before. However, it should be noted that several authors have come close to discovering the links between aeroelasticity and

multiparameter spectral theory. Bisplinghoff [3], in his seminal textbook, describes flutter as a "double eigenvalue problem, where two characteristic numbers determine the [flutter] speed and frequency", and this is an assertion that is repeated in a number of sources, including the latest MSC Nastran aeroelastic analysis user's guide [14]. However, the actual implications of this phrase – that the eigenvalue of the aeroelastic system was not simply the flutter frequency, but a 2-tuple of flutter speed and frequency – remained unexplored. In the case of Bisplinghoff (writing in 1955) this is quite understandable, because many of the key theoretical results in multiparameter spectral theory were only developed by Atkinson the late 1960s [47,48], and the first applications of this theory to real-world systems have occurred only recently [49–51].

We would thus prefer not to class our analysis into any one of the existing aeroelastic methods. We will be using Eq. 1.1 as a stability criterion, and in some sense our work can be most closely related to classical flutter analysis. However, we will also show how these models can be used to analyse subcritical and supercritical phenomena – usually considered the domain of the p-method. And our analysis applies equally well to methods formulated according to the p-k method, k-method, and the all the various variants thereof. We will not consider the possible applications of our analysis to other more novel methods – such as the μ-type methods – but at the very least there is the potential for applicability here. The example systems that will be used to demonstrate our methods are drawn from the study of classical flutter. However, it is important to note that this is for convenience only, and our techniques are applicable to a wide range of aeroelastic models.

1.2 CENTRAL THESIS

The central concept of this thesis is the idea that the problem of solving an aeroelastic system for its flutter points is nothing other than a multiparameter eigenvalue problem. Consider a linear system with eigenvector $\mathbf{x}$ and arbitrary continuous dependence on both an eigenvalue parameter $\chi \in \mathbb{C}$, and another structural or environmental parameter $p \in \mathbb{R}$:

$$\mathrm{A}(\chi, p)\mathbf{x} = 0 \tag{1.2.1}$$

where $\mathrm{A} \in \mathbb{C}^{n\times n}$. Any complex structural parameter can of course be split into two real parameters. For now we consider the case where the system is finite-dimensional. The

stability problem for this system (with respect to parameter p) is to find $\chi, p : [\mathrm{Im}(\chi) = 0, A(\chi, p)\mathbf{x} = 0]$; that is, p such that an eigenvalue of the problem (χ) has zero imaginary part. This point is the 'stability boundary': for a system with multiple structural parameters, the stability boundary may be a line or another higher-dimensional surface.

We then note that the condition $\mathrm{Im}(\chi) = 0$ is equivalent to modifying the original definition of the problem such that $\chi \in \mathbb{R}$ and not $\chi \in \mathbb{C}$. This modification is implicit in classical flutter analysis, but is not usually articulated in this sense[5]. However, such a manoeuvre does not seem to be immediately useful. Under $\chi \in \mathbb{R}$, a solution to Eq. 1.2.1 only exists on the stability boundary, and nowhere else. In order to develop, for example, iterative methods for flutter point calculation, we really need to be able to define some form of solution in the subcritical and supercritical areas (above and below the stability boundary, respectively). There is an easy way of doing this. The method has been applied before in the stability analysis of delay differential equations [50], but has never been used in aeroelasticity or any other structural stability problem that the author is aware of. We take the complex conjugate of Eq. 1.2.1 and add it as another equation:

$$\begin{aligned} \mathrm{A}(\chi, p)\mathbf{x} &= 0 \\ \overline{\mathrm{A}}(\chi, p)\bar{\mathbf{x}} &= 0 \end{aligned} \tag{1.2.2}$$

One may ask what information this conjugation operation adds to the system – surely $\mathrm{A}(\chi, p)\mathbf{x} = 0$ implies $\overline{\mathrm{A}}(\chi, p)\bar{\mathbf{x}} = 0$? This is not true: χ and p are unaffected by the conjugation, as they are real. The conjugation operation thus encodes the information that $p \in \mathbb{R}$ and $\chi \in \mathbb{R}$. Equation 1.2.2 is nothing other than a multiparameter eigenvalue problem: an eigenvalue problem in which the eigenvalue point is not simply defined by a scalar and an eigenvector, but by an n-tuple and an eigenvector. A number of methods of analysis have been developed for such problems, and it will be the task of this thesis to apply these methods to the stability analysis of aeroelastic structures, and to develop new multiparameter solution methods that are tailored specifically to aeroelastic applications.

[5] Classical flutter analysis attempts to solve the system only for its flutter points (without considering subcritical or supercritical behaviour); and ignores the possibility that it is still possible to consider the dependence of Eq. 1.2.1 on $\chi \in \mathbb{R}$, which we will show can give information on subcritical and supercritical behaviour

For, as we will see, the solution methods that are available for Eq. 1.2.2 depend strongly on the structure of matrix A.

1.3 EXTENSIONS

If we can get over the conceptual hurdle of treating the airspeed parameter in an aeroelastic equation as an eigenvalue, then it is easy to see that there is no barrier to us treating *any* model parameters in such an equation as an eigenvalue. This opens up a massive field of possibilities.

As a start, we might consider changing our eigenvalue selections in our simple χ-p system. For example, given a system that depends on modal frequency χ, airspeed parameter p, and also (say) a mass parameter m and stiffness parameter k. We could solve the system for a its flutter points at a given mass and stiffness (eigenvalues χ and p), or for a system of known stiffness we could solve for the mass that causes a flutter point to be at a given location (eigenvalues χ and m). Or, given the location of a flutter point, we could look at the possible systems that could generate such a flutter point (eigenvalues m and k). From this we could deduce the number of flutter point locations that must be known in order to identify the model. Both of the latter two eigenvalue choices could be very useful in model identification.

But perhaps more interestingly, we could also start looking at higher-dimensional systems, with not just two but n parameters. We might look at extending the definition of the flutter point to include the effect of flight altitude / air density / Mach number (all of these parameters model the same phenomenon). We could then compute points on the aeroelastic flight envelope of an aircraft. Also, using the same rationale as in our mass-stiffness example, we could compute the set of model parameters that would be required to move points on the flight envelope to different locations. We note that any scalar equation whatsoever may be converted into a multiparameter eigenvalue problem: if we write the equation as $G(\boldsymbol{\lambda}) = 0$, for residual G and vector of eigenvalues $\boldsymbol{\lambda}$, then it follows that $G(\boldsymbol{\lambda})x = 0$ for an arbitrary scalar eigenvector $x \neq 0$. Of course, the introduction of this eigenvector is redundant, but necessary if we wish to treat the equation $G(\boldsymbol{\lambda}) = 0$ as part of our eigenproblem system. Though we do not go that far in this work, it may be possible to

derive hybrid multiparameter solvers which treat a set of multiparameter eigenvalue problems coupled with a set of non-eigenproblem constraint equations. It is probably even possible to develop a multiparameter eigenproblem least-squares approach for solving multiparameter eigenvalue problems which are overconstrained (i.e. have more distinct equations than parameters).

In this thesis we will be primarily concerned with the standard two-parameter multiparameter eigenvalue problems of Section 1.2, and will omit a treatment of these more advanced problems. However, when developing multiparameter solvers, we will consider their prospects for extension into higher dimensions. We will find that most of the methods that we develop are easy to extend in this way. We will also return to the future prospects of multiparameter aeroelastic solvers, and the avenues for future research in this area, in Chapter 7.

1.4 REFERENCES

[1] Garrick, I. E., and Reed III, W. H., 1981, "Historical development of aircraft flutter," J. Aircr., **18**(11), pp. 897–912.

[2] Hodges, D. H., and Pierce, G. A., 2011, Introduction to Structural Dynamics and Aeroelasticity, Cambridge University Press, New York, New York State, USA.

[3] Bisplinghoff, R. L., Ashley, H., and Halfman, R. L., 1957, Aeroelasticity, Addison-Wesley, Reading, Massachusetts, USA.

[4] Dowell, E. H., 1975, Aeroelasticity of plates and shells, Noordhoff International Publishing, Leyden, The Netherlands.

[5] Fung, Y. C., 2002, An Introduction to the Theory of Aeroelasticity, Dover Publications, New York, New York State, USA.

[6] Li, D., Guo, S., and Xiang, J., 2010, "Aeroelastic dynamic response and control of an airfoil section with control surface nonlinearities," J. Sound Vib., **329**(22), pp. 4756–4771.

[7] Lee, B. H. K., Gong, L., and Wong, Y. S., 1997, "Analysis and computation of nonlinear dynamic response of a two-degree-of-freedom system and its application in aeroelasticity," J. Fluids Struct., **11**(3), pp. 225–246.

[8] Bartholomew, P., 1998, “The role of MDO within aerospace design and progress towards an MDO capability,” AIAA, St. Louis, MO, U.S.A., pp. 98–4705.

[9] Pendleton, E. W., Bessette, D., Field, P. B., Miller, G. D., and Griffin, K. E., 2000, “Active aeroelastic wing flight research program: technical program and model analytical development,” J. Aircr., **37**(4), pp. 554–561.

[10] Al-Assaf, A., 2006, “Flutter analysis of open-truss stiffened suspension bridges using synthesized aerodynamic derivatives,” Doctoral Dissertation, Washington State University.

[11] Namini, A., Albrecht, P., and Bosch, H., 1992, “Finite element-based flutter analysis of cable-suspended bridges,” J. Struct. Eng., **118**(6), pp. 1509–1526.

[12] Larsen, A., 1998, “Advances in aeroelastic analyses of suspension and cable-stayed bridges,” J. Wind Eng. Ind. Aerodyn., **74**, pp. 73–90.

[13] Salvatori, L., and Borri, C., 2007, “Frequency-and time-domain methods for the numerical modeling of full-bridge aeroelasticity,” Comput. Struct., **85**(11), pp. 675–687.

[14] Rodden, W. P., and Johnson, E. H., 2004, MSC.Nastran Aeroelastic Analysis, User’s Guide - Version 68, MacNeal-Schwendler Corporation.

[15] ZONA Technology, Inc., 2011, “ZAERO Theoretical Manual, Version 8.5.”

[16] Murua, J., Palacios, R., and Peiró, J., 2010, “Camber effects in the dynamic aeroelasticity of compliant airfoils,” J. Fluids Struct., **26**(4), pp. 527–543.

[17] Palacios, R., and Cesnik, C. E. S., 2008, “On the one-dimensional modeling of camber bending deformations in active anisotropic slender structures,” Int. J. Solids Struct., **45**(7-8), pp. 2097–2116.

[18] Loewy, R. G., 1957, “A two-dimensional approximation to the unsteady aerodynamics of rotary wings,” J. Aeronaut. Sci. Inst. Aeronaut. Sci., **24**(2).

[19] Shipman, K. W., and Wood, E. R., 1971, “A two-dimensional theory for rotor blade flutter in forward flight,” J. Aircr., **8**(12), pp. 1008–1015.

[20] Rauchenstein Jr., W. J., 2002, “A 3D Theodorsen-based rotor blade flutter model using normal modes,” Master’s Thesis, Naval Postgraduate School.

[21] Pedley, T. J., and Hill, S. J., 1999, “Large-amplitude undulatory fish swimming: fluid mechanics coupled to internal mechanics,” J. Exp. Biol., **202**(23), pp. 3431–3438.

[22] Liu, T., Kuykendoll, K., Rhew, R., and Jones, S., 2006, “Avian wing geometry and kinematics,” AIAA J., **44**(5), pp. 954–963.

[23] Ansari, S. A., Żbikowski, R., and Knowles, K., 2006, "Aerodynamic modelling of insect-like flapping flight for micro air vehicles," Prog. Aerosp. Sci., **42**(2), pp. 129–172.

[24] Frazer, R. A., and Duncan, W. J., 1928, The Flutter of Aeroplane Wings, British A. R. C.

[25] Frazer, R. A., and Duncan, W. J., 1931, The Flutter of Monoplanes, Biplanes and Tail Units: A Sequel to R. & M. 1155, HM Stationery Office, London, UK.

[26] Lawrence, A. J., and Jackson, P., 1970, Comparison of Different Methods of assessing the Free Oscillatory Characteristics of Aeroelastic System, British Aeronautical Research Council, London, UK.

[27] Hassig, H. J., 1971, "An approximate true damping solution of the flutter equation by determinant iteration.," J. Aircr., **8**(11), pp. 885–889.

[28] Lind, R., and Brenner, M., 1999, Robust aeroservoelastic stability analysis: Flight test applications, Springer-Verlag London, London, UK.

[29] Lind, R., 2002, "Match-Point Solutions for Robust Flutter Analysis," J. Aircr., **39**(1), pp. 91–99.

[30] Borglund, D., 2003, "Robust aeroelastic stability analysis considering frequency-domain aerodynamic uncertainty," J. Aircr., **40**(1), pp. 189–193.

[31] Borglund, D., 2004, "The mu-k method for robust flutter solutions," J. Aircr., **41**(5), pp. 1209–1216.

[32] Borglund, D., 2005, "Upper Bound Flutter Speed Estimation Using the mu-k Method," J. Aircr., **42**(2), pp. 555–557.

[33] Borglund, D., and Ringertz, U., 2006, "Efficient computation of robust flutter boundaries using the mu-k method," J. Aircr., **43**(6), pp. 1763–1769.

[34] Gu, Y., Yang, Z., Wang, W., and Xia, W., 2009, "Dynamic Pressure Perturbation Method for Flutter Solution: the Mu-Omega Method," Palm Springs, California, USA.

[35] Gu, Y., and Yang, Z., 2010, "Generalized Mu-Omega Method with Complex Perturbation to Dynamic Pressure," Proceedings of the 51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Orlando, Florida, USA.

[36] Borglund, D., 2007, "Robust Aeroelastic Analysis in the Laplace Domain: The μ-p Method," Proceedings of the 2007 International Forum on Aeroelasticity and Structural Dynamics (IFASD), Stockholm, Sweden.

[37] Afolabi, D., 1994, “Flutter analysis using transversality theory,” Acta Mech., **103**(1-4), pp. 1–15.

[38] Afolabi, D., Pidaparti, R. M. V., and Yang, H. T. Y., 1998, “Flutter Prediction Using an Eigenvector Orientation Approach,” AIAA J., **36**(1), pp. 69–74.

[39] Gu, Y., Zhang, X., and Yang, Z., 2012, “Robust flutter analysis based on genetic algorithm,” Sci. China Technol. Sci., **55**(9), pp. 2474–2481.

[40] Blythe, P. W., and Herszberg, I. H., 1993, “The Solution of Flutter Equations Using Neural Networks,” 5th Australian Aeronautical Conference: Preprints of Papers, Institution of Engineers, Australia, p. 415.

[41] Chen, C. H., 2003, “Determination of flutter derivatives via a neural network approach,” J. Sound Vib., **263**(4), pp. 797–813.

[42] Chen, C.-H., Wu, J.-C., and Chen, J.-H., 2008, “Prediction of flutter derivatives by artificial neural networks,” J. Wind Eng. Ind. Aerodyn., **96**(10), pp. 1925–1937.

[43] Natarajan, A., 2002, “Aeroelasticity of morphing wings using neural networks,” Doctoral Dissertation, Virginia Polytechnic Institute and State University.

[44] Haddadpour, H., and Firouz-Abadi, R. D., 2009, “True damping and frequency prediction for aeroelastic systems: The PP method,” J. Fluids Struct., **25**(7), pp. 1177–1188.

[45] Chen, P. C., 2000, “Damping perturbation method for flutter solution: the g-method,” AIAA J., **38**(9), pp. 1519–1524.

[46] Irani, S., and Sazesh, S., 2013, “A new flutter speed analysis method using stochastic approach,” J. Fluids Struct., **40**, pp. 105–114.

[47] Atkinson, F. V., 1968, “Multiparameter spectral theory,” Bull. Am. Math. Soc., **74**(1), pp. 1–28.

[48] Atkinson, F. V., 1972, Multiparameter Eigenvalue Problems: Matrices and Compact Operators, Academic Press, London, UK.

[49] Molzahn, D., 2010, “Power system models formulated as eigenvalue problems and properties of their solutions,” Master’s Thesis, University of Wisconsin–Madison.

[50] Meerbergen, K., Schröder, C., and Voss, H., 2013, “A Jacobi-Davidson method for two-real-parameter nonlinear eigenvalue problems arising from delay-differential equations,” Numer. Linear Algebra Appl., **20**(5), pp. 852–868.

[51] Jarlebring, E., and Hochstenbach, M. E., 2009, “Polynomial two-parameter eigenvalue problems and matrix pencil methods for stability of delay-differential equations,” Linear Algebra Its Appl., **431**(3), pp. 369–380.

Chapter 2
Index of systems

2.1 INTRODUCTION

In the previous chapter we introduced the link between multiparameter spectral theory and aeroelastic flutter. In the following chapters we will devise multiparameter methods for solving flutter problems. However, before this can be done we must define the types of flutter problems that we will be working with, and show specifically how they can be expressed as multiparameter eigenvalue problems. This is the purpose of the current chapter. We will focus on small-scale discrete problems, as a way of developing and exploring new multiparameter methods – and as we develop these methods, we will discuss their applicability to larger problems.

We should note that our treatment of aeroelastic flutter problems is by no means comprehensive, but is only intended to be indicative of the types of aeroelastic problems that could be solved via a multiparameter analysis. In later chapters, when we devise multiparameter solution methods, our approach will be both specific and general: we will devise methods for general classes of problems – e.g. polynomial, polynomial with black-box scalar function, matrix black-box function – and then apply these methods to aeroelastic problems with this form. At the coarsest level, we will classify problems into three classes: structured, semistructured and unstructured problems. Structured problems, under our definition, have a fully analytical structure – e.g. polynomial problems. For such problems we expect direct solution methods to exist. Unstructured problems, on the other hand, have no analytical structure, and consist entirely of a black-box matrix function. Such problems will almost exclusively be solved by iterative solvers. And semistructured problems – perhaps the most interesting class – are those problems that contain both structured and unstructured elements – e.g. a polynomial problem also containing a black-box scalar function. We will find that we can devise both iterative and direct solvers for such problems. This classification is entirely our own invention and is not found in existing literature – but this is probably because no semistructured or unstructured multiparameter problems have

ever been considered before; neither in theoretical nor applied literature. Our treatment of these problems should thus be interesting both to mathematicians working in the field of multiparameter spectral theory, and to engineers in aeroelasticity and other fields.

2.2 AN INTRODUCTORY MODEL

Consider first the simple section model shown in Figure 2.1.

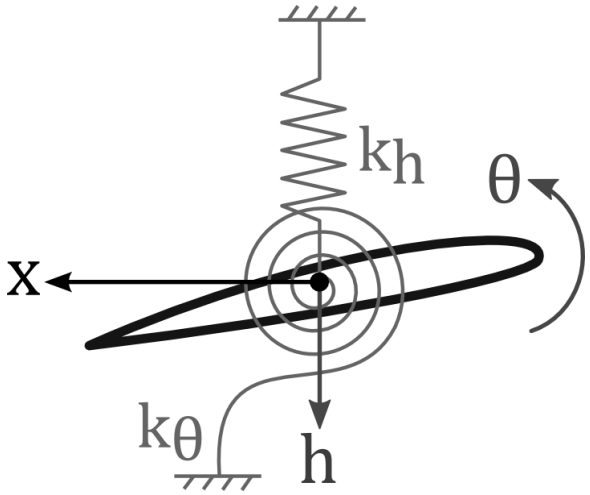


Figure 2.1: A section model without damping.

This model has two degrees of freedom: bending displacement h and twist θ. In aeroelastic literature the bending displacement, when defined downwards, is usually referred to as *plunge*. The governing equations for this model are easy to derive. They are:

$$\begin{aligned} m\ddot{h} + k_h h - mx_\theta \ddot{\theta} &= -L(t) \\ I_P \ddot{\theta} + k_\theta \theta - mx_\theta \ddot{h} &= M(t), \end{aligned} \tag{2.2.1}$$

where m and I_P are the section mass and polar moment of inertia, k_h and k_θ are the section bending and twist stiffnesses, x_θ is the section's static imbalance[1], and $L(t)$ and $M(t)$ are the aerodynamic lift and moment. Using basic steady-flow aerodynamics [1], we have

$$\begin{aligned} L &= 2\pi\rho_\infty b U^2 \theta \\ M &= 2\pi\rho_\infty b^2 U^2 \left(\tfrac{1}{2} + a\right)\theta, \end{aligned} \tag{2.2.2}$$

where U is the airspeed, ρ_∞ is the free-stream air density, b is the semichord length and a is the distance along the x-axis from the leading edge to the centre of mass, as a fraction of

[1] Defined as the distance along the x-axis from the pivot point to the centre of mass.

the semichord. Note that Eq. 2.2.2 assumes that the aerodynamic moment on the aerofoil is produced only by the offset of the lift force (acting at the quarter-chord point) from the pivot point. This is a very simple aerodynamic model which would not be suitable for any real-life aeroelastic analysis, but it will be useful for the purposes of introducing our multiparameter method.

Applying the Fourier transform $[h(t), \theta(t)] = \left[\hat{h}, \hat{\theta}\right] \exp(\iota\chi t)$ to Eq. 2.2.1 yields:

$$\begin{aligned} (-m\chi^2 + k_h)\hat{h} + mx_\theta\chi^2\hat{\theta} &= -2\pi\rho_\infty bU^2\hat{\theta} \\ mx_\theta\chi^2\hat{h} + (-I_P\chi^2 + k_\theta)\hat{\theta} &= 2\pi\rho_\infty b^2U^2\left(\tfrac{1}{2} + a\right)\hat{\theta} \end{aligned} \tag{2.2.3}$$

or

$$\left(\begin{bmatrix} k_h & 0 \\ 0 & k_\theta \end{bmatrix} + \begin{bmatrix} -m & mx_\theta \\ mx_\theta & -I_P \end{bmatrix}\chi^2 + \begin{bmatrix} 0 & 2\pi\rho_\infty b \\ 0 & -2\pi\rho_\infty b^2\left(\frac{1}{2} + a\right) \end{bmatrix} U^2\right)\begin{bmatrix} \hat{h} \\ \hat{\theta} \end{bmatrix} = \mathbf{0} \tag{2.2.4}$$

Defining a new eigenvalue parameter, $\Upsilon = U/b$, and the following dimensionless parameters:

mass ratio $\mu = \dfrac{m}{\rho\pi b^2}$

dimensionless radius of gyration $r = \sqrt{\dfrac{I_P}{mb^2}}$

dimensionless static imbalance $r_\theta = \dfrac{x_\theta}{b}$

uncoupled bending natural frequency $\omega_h = \sqrt{\dfrac{k_h}{m}}$

uncoupled torsional natural frequency $\omega_\theta = \sqrt{\dfrac{k_\theta}{I_P}}$.

Following a multiplication of the whole system by -1, Eq. 2.2.4 becomes

$$\left(\begin{bmatrix} -\omega_h^2 & 0 \\ 0 & -r^2\omega_\theta^2 \end{bmatrix} + \begin{bmatrix} 1 & -r_\theta \\ -r_\theta & r^2 \end{bmatrix}\chi^2 + \frac{1}{\mu}\begin{bmatrix} 0 & -2 \\ 0 & 2\left(\frac{1}{2} + a\right) \end{bmatrix}\Upsilon^2\right)\begin{bmatrix} \hat{h}/b \\ \hat{\theta} \end{bmatrix} = \mathbf{0}. \tag{2.2.5}$$

The parameter Υ can be interpreted physically as the airspeed measured in semichords per second. Eq. 2.2.5 is of the form

$$(\mathrm{A} + \mathrm{B}\chi^2 + \mathrm{C}\Upsilon^2)\mathbf{x} = 0. \tag{2.2.6}$$

which is one equation of a two-parameter eigenvalue problem in $\lambda = \chi^2$ and $\mu = \Upsilon^2$. As shown in Chapter 1, we can define the second equation as the complex conjugate of Eq. 2.2.5:

$$\begin{aligned}(\mathrm{A} + \mathrm{B}\lambda + \mathrm{C}\mu)\mathbf{x} = 0,\\ (\overline{\mathrm{A}} + \overline{\mathrm{B}}\lambda + \overline{\mathrm{C}}\mu)\overline{\mathbf{x}} = 0.\end{aligned} \tag{2.2.7}$$

This is now a multiparameter eigenvalue problem. It is in fact a special kind of multiparameter eigenvalue problem; namely, a linear one. The numerical parameters for this model are chosen to match those from an identical (but nondimensional) system in [1]. These parameters are shown in Table 2.1. Table 2.2 shows one set of equivalent dimensional parameters which can be used in Eq. 2.2.4.

Table 2.1: Dimensionless parameter values for Eq. 2.5

Parameter	Value
mass ratio – μ	20
radius of gyration – r	0.4899
bending nat. freq. – ω_h	0.5642 rad/s
torsional nat. freq. – ω_θ	1.4105 rad/s
static imbalance – r_θ	-0.1
centre of mass location – a	-0.2

Table 2.2: Equivalent dimensional parameter values

Parameter	Value
mass – m	20π kg
rotational inertia – I	4.8π kgm^2
bending stiffness – k_h	30 N/m
torsional stiffness – k_θ	20 Nm/rad
static imbalance – x_θ	-0.1 m
semichord – b	1 m
centre of mass location – a	-0.2

From [1] we know that this model has one divergence point and one flutter point. The divergence point can be computed analytically: it lies at

$$\Upsilon_D = \omega_\theta r\sqrt{\frac{\mu}{1+2a}} = 3.990 \text{ Hz}. \tag{2.2.8}$$

The flutter point has been computed by Hodges and Pierce [1] to lie at $\Upsilon_F = 2.600$ Hz and $\chi_F = 0.7853$ rad/s.

2.3 A SECTION MODEL WITH UNSTEADY AERODYNAMICS

2.3.1 Formulation in χ and k

The major failing of the previous model – in terms of modelling real-life aeroelastic behaviour – is its aerodynamic model, which does not account for unsteady effects. We therefore derive a model which includes a more accurate representation of these effects. We also modify the structural model, introducing bending and twist damping. Figure 2.2 shows the new structural model.

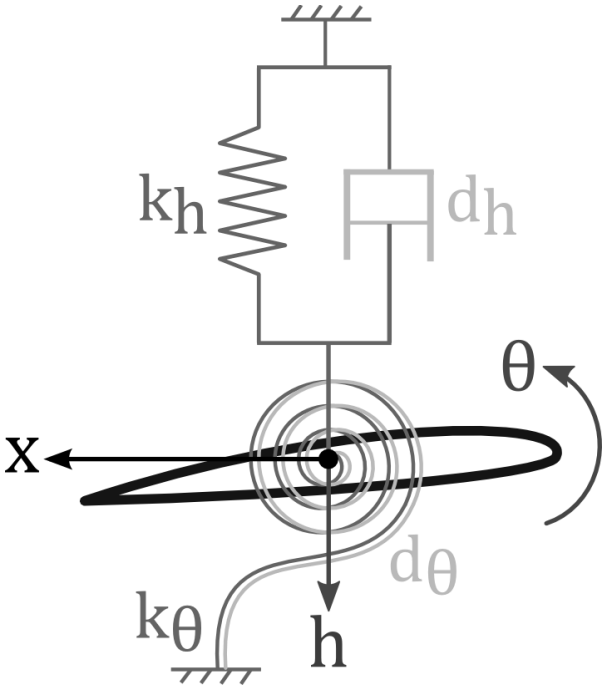


Figure 2.2: Section model with structural damping

The governing equations for this model are:

$$\begin{aligned} m\ddot{h} + d_h\dot{h} + k_h h - mx_\theta\ddot{\theta} &= -L(t) \\ I_P\ddot{\theta} + d_\theta\dot{\theta} + k_\theta\theta - mx_\theta\ddot{h} &= M(t), \end{aligned} \tag{2.3.1}$$

where m and I_P are the section mass and polar moment of inertia, k_h and k_θ are the section bending and twist stiffnesses, d_h and d_θ are the section bending and twist damping coefficients, x_θ is the section's static imbalance[2], and $L(t)$ and $M(t)$ are the aerodynamic lift and moment. We will be concerned with the frequency domain stability analysis of this system, and thus it is necessary for us to express the unsteady loads L and M in the frequency domain. To do this we use Theodorsen's unsteady aerodynamic theory [1]:

[2] Defined as the distance along the x-axis from the pivot point to the centre of mass.

$$L = -\chi^2\left(L_h\hat{h} + L_\theta\hat{\theta}\right)$$
$$M = \chi^2\left(M_h\hat{h} + M_\theta\hat{\theta}\right). \tag{2.3.2}$$

Note that we have again defined our eigenvalue χ such that $[h(t), \theta(t)] = \left[\hat{h}, \hat{\theta}\right]\exp(\iota\chi t)$. The aerodynamic coefficients $\{L_h, L_\theta, M_h, M_\theta\}$ are given by:

$$L_h(\kappa) = \pi\rho b^2\left(1 - \frac{2\iota C(\kappa)}{\kappa}\right),$$
$$L_\theta(\kappa) = \pi\rho b^3\left(-a - \frac{\iota}{\kappa} - \frac{2C(\kappa)}{\kappa^2} - \frac{2\iota\left(\frac{1}{2} - a\right)C(\kappa)}{\kappa}\right),$$
$$M_h(\kappa) = \pi\rho b^3\left(-a + \frac{2\iota\left(\frac{1}{2} + a\right)C(\kappa)}{\kappa}\right), \tag{2.3.3}$$
$$M_\theta(\kappa) = \pi\rho b^4\left(\frac{1}{8} + a^2 - \frac{\iota\left(\frac{1}{2} - a\right)}{\kappa} + \frac{2\left(\frac{1}{2} + a\right)C(\kappa)}{\kappa^2} + \frac{2\iota\left(\frac{1}{4} - a^2\right)C(\kappa)}{\kappa}\right),$$

where κ is the *reduced frequency*[3], a widely-used parameter in aeroelasticity:

$$\kappa = \frac{b\chi}{U}, \tag{2.3.4}$$

and a is the distance along the x-axis from the leading edge to the centre of mass, as a fraction of the semichord. $C(\kappa)$ is Theodorsen's function

$$C(\kappa) = \frac{H_1^{(2)}(\kappa)}{H_1^{(2)}(\kappa) - \iota H_0^{(2)}(\kappa)}, \tag{2.3.5}$$

where $H_n^{(2)}$ is the n-th Hankel function of the second kind [2]. The aerodynamic coefficients in Eq. 2.3.3 assume a lift-angle of attack coefficient of $C_{L\alpha} = 2\pi$, as per standard thin-airfoil theory [3]: modifications can be made to these coefficients to account for an arbitrary lift-angle of attack coefficient $C_{L\alpha} \neq 2\pi$. However, as these modifications do not change the form of the equations, we take $C_{L\alpha} = 2\pi$ without loss of generality.

[3] Note: this is typically given the symbol k in most aeroelastic literature. k will be necessary as an index or iteration counter when we go on to develop solution algorithms for these flutter systems, and so here we use κ.

Applying the same Fourier transform used in the aerodynamic loads, $[h(t), \theta(t)] = \left[\hat{h}, \hat{\theta}\right] \exp(\iota\chi t)$, to Eq. 2.3.1 yields:

$$\begin{aligned}(-m\chi^2 + \iota d_h\chi + k_h)\hat{h} + mx_\theta\chi^2\hat{\theta} &= \chi^2\left(L_h\hat{h} + L_\theta\hat{\theta}\right)\\ mx_\theta\chi^2\hat{h} + (-I_P\chi^2 + \iota d_\theta\chi + k_\theta)\hat{\theta} &= \chi^2\left(M_h\hat{h} + M_\theta\hat{\theta}\right)\end{aligned} \tag{2.3.6}$$

or

$$\left(\begin{bmatrix}-m - L_h(\kappa) & mx_\theta - L_\theta(\kappa)\\ mx_\theta - M_h(\kappa) & -I_P - M_\theta(\kappa)\end{bmatrix}\chi^2 + \begin{bmatrix}\iota d_h & 0\\ 0 & \iota d_\theta\end{bmatrix}\chi + \begin{bmatrix}k_h & 0\\ 0 & k_\theta\end{bmatrix}\right)\begin{bmatrix}\hat{h}\\ \hat{\theta}\end{bmatrix} = \mathbf{0} \tag{2.3.7}$$

This is a generalised eigenproblem in χ^2 dependent on the parameter κ. After multiplying the whole system by -1 and rearranging, we obtain

$$\left(\left(\mathrm{M}_0^* + \mathrm{G}_0^* + \mathrm{G}_1^*\frac{1}{\kappa} + \mathrm{G}_2^*\frac{C(\kappa)}{\kappa} + \mathrm{G}_3^*\frac{C(\kappa)}{\kappa^2}\right)\chi^2 - \mathrm{D}_0^*\chi - \mathrm{K}_0^*\right)\mathbf{x}^* = \mathbf{0}. \tag{2.3.8}$$

where $\mathbf{x} = \left[\hat{h}; \hat{\theta}\right]$ and

$$\mathrm{G}_0^* = \begin{bmatrix}\pi\rho b^2 & -a\pi\rho b^3\\ -a\pi\rho b^3 & \pi\rho b^4\left(\frac{1}{8} + a^2\right)\end{bmatrix},\ \mathrm{G}_1^* = \begin{bmatrix}0 & -\iota\pi\rho b^3\\ 0 & -\iota\pi\rho b^4\left(\frac{1}{2} - a\right)\end{bmatrix},$$

$$\mathrm{G}_2^* = \begin{bmatrix}-2\iota\pi\rho b^2 & -2\iota\left(\frac{1}{2} - a\right)\pi\rho b^3\\ 2\iota\left(\frac{1}{2} + a\right)\pi\rho b^3 & 2\iota\left(\frac{1}{4} - a^2\right)\pi\rho b^4\end{bmatrix},\ \mathrm{G}_3^* = \begin{bmatrix}0 & -2\pi\rho b^3\\ 0 & 2\left(\frac{1}{2} + a\right)\pi\rho b^4\end{bmatrix}, \tag{2.3.9}$$

$$\mathrm{M}_0^* = \begin{bmatrix}m & -mx_\theta\\ -mx_\theta & I_p\end{bmatrix},\qquad \mathrm{D}_0^* = \begin{bmatrix}\iota d_h & 0\\ 0 & \iota d_\theta\end{bmatrix},\qquad \mathrm{K}_0^* = \begin{bmatrix}k_h & 0\\ 0 & k_\theta\end{bmatrix},$$

This gives a much better idea of the structure of A with respect to the reduced frequency κ. Defining the following dimensionless (and near-dimensionless) parameters

mass ratio	$\mu = \dfrac{m}{\rho\pi b^2}$
dimensionless radius of gyration	$r = \sqrt{\dfrac{I_P}{mb^2}}$
dimensionless static imbalance	$r_\theta = \dfrac{x_\theta}{b}$

uncoupled bending natural frequency $\omega_h = \sqrt{\frac{k_h}{m}}$

uncoupled torsional natural frequency $\omega_\theta = \sqrt{\frac{k_\theta}{I_P}}$

uncoupled bending damping ratio $\zeta_h = \frac{1}{2}\frac{d_h}{\sqrt{mk_h}}$

uncoupled torsional damping ratio $\zeta_\theta = \frac{1}{2}\frac{d_\theta}{\sqrt{I_P k_\theta}}$

we can write Eq. 2.3.8 as:

$$\left(\left(\mathrm{M}_0 + \mathrm{G}_0 + \mathrm{G}_1\frac{1}{\kappa} + \mathrm{G}_2\frac{C(\kappa)}{\kappa} + \mathrm{G}_3\frac{C(\kappa)}{\kappa^2}\right)\chi^2 - \mathrm{D}_0\chi - \mathrm{K}_0\right)\mathbf{x} = \mathbf{0}. \tag{2.3.10}$$

with a nondimensional eigenvector $\mathbf{x} = \left[\hat{h}/b\,;\hat{\theta}\right]$ and the matrix coefficients

$$\mathrm{G}_0 = \frac{1}{\mu}\begin{bmatrix} 1 & -a \\ -a & \left(\frac{1}{8}+a^2\right)\end{bmatrix}, \qquad \mathrm{G}_1 = \frac{1}{\mu}\begin{bmatrix} 0 & -\iota \\ 0 & -\iota\left(\frac{1}{2}-a\right)\end{bmatrix},$$

$$\mathrm{G}_2 = \frac{1}{\mu}\begin{bmatrix} -2\iota & -2\iota\left(\frac{1}{2}-a\right) \\ 2\iota\left(\frac{1}{2}+a\right) & 2\iota\left(\frac{1}{4}-a^2\right)\end{bmatrix}, \qquad \mathrm{G}_3 = \frac{1}{\mu}\begin{bmatrix} 0 & -2 \\ 0 & 2\left(\frac{1}{2}+a\right)\end{bmatrix}, \tag{2.3.11}$$

$$\mathrm{M}_0 = \begin{bmatrix} 1 & -r_\theta \\ -r_\theta & r^2\end{bmatrix}, \qquad \mathrm{D}_0 = \begin{bmatrix} 2\iota\zeta_h\omega_h & 0 \\ 0 & 2\iota r^2\zeta_\theta\omega_\theta\end{bmatrix}, \qquad \mathrm{K}_0 = \begin{bmatrix} \omega_h^2 & 0 \\ 0 & r^2\omega_\theta^2\end{bmatrix}.$$

This form is less complex, and has the additional advantage than the nondimensional parameters that have been defined are widely used in aeroelastic literature. Note that Eq. 2.3.10 is of exactly the same form as Eq. 2.3.8. The analysis that we will perform does not rely on the nondimensionalisation – this is simply for convenience. However, Eq. 2.3.10 is not currently in a useful form for multiparameter analysis, consisting as it does of both inverted polynomial ($1/\kappa$) and polynomial (χ) eigenvalue functions. Ideally, we would like a fully polynomial system – apart, of course, from Theodorsen's function. Simply multiplying out by κ^2 is unsatisfactory, as it results in terms of the order $\kappa^2\chi^2$ (i.e. fourth order) when in fact the system only needs to be second-order. We should thus seek to find other formulations of this problem which are second-order and polynomial.

2.3.2 Other formulations

Separating Eq. 2.3.10 out into its physical, dimensional eigenvalue parameters (U and χ), we obtain

$$\left(\left(\mathrm{M}_0+\mathrm{G}_0+\mathrm{G}_1\frac{U}{b\chi}+\mathrm{G}_2C\left(\frac{b\chi}{U}\right)\frac{U}{b\chi}+\mathrm{G}_3C\left(\frac{b\chi}{U}\right)\frac{U^2}{b^2\chi^2}\right)\chi^2-\mathrm{D}_0\chi-\mathrm{K}_0\right)\mathbf{x}=\mathbf{0}. \quad (2.3.12)$$

Defining a new eigenvalue parameter, $\Upsilon = U/b$, which can be interpreted physically as the airspeed measured in semichords per second, we obtain the eigenvalue problem

$$\left((\mathrm{M}_0+\mathrm{G}_2)\chi^2+\left(\mathrm{G}_1+\mathrm{G}_2C\left(\frac{\chi}{\Upsilon}\right)\right)\Upsilon\chi+\mathrm{G}_3C\left(\frac{\chi}{\Upsilon}\right)\Upsilon^2-\mathrm{D}_0\chi-\mathrm{K}_0\right)\mathbf{x}=\mathbf{0}. \quad (2.3.13)$$

We will terms the parameter Υ the *air frequency*, and thus Eq. 2.3.13 is the Υ-χ form of the section model with Theodorsen aerodynamics. We will use this form often in the following chapters. Its main advantage over Eq. 2.3.1 the fact that it is second-order polynomial, apart from $C(\chi/\Upsilon)$. Its main disadvantage is that Theodorsen's function is no longer a function of a single eigenvalue parameter, which also affects our solution method. If we wish to devise a second-order polynomial representation where $C(b\chi/U)$ remains a function of a single variable, we have to define the parameters

$$\tau=\frac{1}{\kappa}=\frac{U}{b\chi},\qquad \lambda=\frac{1}{\chi},\qquad C_\tau(\tau)=C\left(\frac{1}{\tau}\right)=C(\kappa). \quad (2.3.14)$$

which, when substituted, yield the equation:

$$\Big((\mathrm{M}_0+\mathrm{G}_0)+\big(\mathrm{G}_1+\mathrm{G}_2C_\tau(\tau)\big)\tau+\mathrm{G}_3C_\tau(\tau)\tau^2-\mathrm{D}_0\lambda-\mathrm{K}_0\lambda^2\Big)\mathbf{x}=\mathbf{0}. \quad (2.3.15)$$

Both of these new systems are preferable to Eq. 2.3.10, and we will discuss their relative advantages and disadvantages in Chapter 5, when we devise iterative solution methods for both of them. We should note that there are of course dimensional equivalents of Eq. 2.3.15 and 2.3.13: one only needs add the $(\cdot)^*$ and use the matrices defined in Eq. 2.3.9. It should also be noted that, while in this section we will consider these equations in the 2×2 case, they can also be found with larger dimensions: for example, when multiple sections are coupled together to form a full-wing model.

2.3.3 Quasisteady aerodynamics

As they stand, Eq. 2.3.15 and 2.3.13 are effectively only semi-structured. Theodorsen's function is sufficiently complicated (cf. Eq. 2.3.5) that it is far more effective to treat it as a numerical black-box function than to attempt to derive any purely analytical results. This necessitates that the solution methods involved will be iterative. However, in some cases (or with some assumptions) it is possible to reduce these equations to something more tractable. One method is to note that $C(\kappa) \to 1$ as $\kappa \to 0$, and so for low κ it is reasonable to assume $C(\kappa) = 1$ always. Physically, this corresponds to neglecting the effect of the aerofoil's wake vortices – an assumption that is valid only for quasi-steady flow, when the wing is deforming slowly [1,4]. Equation 2.3.13 and 2.3.15 then become polynomial:

$$\left((\mathrm{M}_0 + \mathrm{G}_0) + (\mathrm{G}_1 + \mathrm{G}_2)\tau + \mathrm{G}_3\tau^2 - \mathrm{D}_0\lambda - \mathrm{K}_0\lambda^2\right)\mathbf{x} = \mathbf{0}, \tag{2.3.16}$$

$$\left((\mathrm{M}_0 + \mathrm{G}_0)\chi^2 + (\mathrm{G}_1 + \mathrm{G}_2)\Upsilon\chi + \mathrm{G}_3\Upsilon^2 - \mathrm{D}_0\chi - \mathrm{K}_0\right)\mathbf{x} = \mathbf{0}, \tag{2.3.17}$$

A number of direct solution methods can be derived for Eq. 2.3.16 and Eq. 2.3.17: we discuss these methods in Chapter 4.

2.3.4 Quasisteady aerodynamics with no structural damping

In many aeroelastic systems structural damping is negligible, in which case D_0 is the zero matrix. While this may seem to be a trivial case of the preceeding systems, the omission of the structural damping term does allow us to change the structure of the system, leading to a faster computation time – as we will show later. In the case of Eq. 2.3.16, the omission of structural damping means the system can be written as

$$\left((\mathrm{M}_0 + \mathrm{G}_0) + (\mathrm{G}_1 + \mathrm{G}_2)\tau + \mathrm{G}_3\tau^2 - \mathrm{K}_0\Lambda\right)\mathbf{x} = \mathbf{0}, \tag{2.3.16}$$

where $\Lambda = \lambda^2$. This system is now linear in the parameter Λ, whereas it was previously quadratic. The benefits of this will be seen in Chapter 4. Note that the τ-χ form is the only form we have presented which becomes linear in one variable when the system is undamped – this does not occur with the Υ-χ form.

2.3.5 Approximations to Theodorsen's function

Rather than replacing Theodorsen's function with a single limiting value, we might instead try to replace it with a simpler function that approximates its behaviour. Many such

approximating functions have been developed: Ref. [5] gives an extensive but by no means exhaustive list. The vast majority of these approximations are rational functions in k, as they have been designed for use in control loops where rational transfer functions are preferable. We will consider two approximations to Theodorsen's function in this work: a rational representation of second order in the numerator and denominator, first given in [6]; and a fractional-order representation given in [7]. The use of the rational approximation will demonstrate that our methods can be used with any rational approximation to Theodorsen's function, and the use of the fractional approximation will show also that our methods can be extended to more complex approximating functions than are widely used at present. We should note that fractional-order multiparameter eigenvalue problems have never been considered before, in neither practical nor theoretical literature. Our treatment of this problem opens up both an interesting area for future abstract research – the study of fractional multiparameter eigenvalue problems – and a wider class of fractional-order systems for use by practical modellers.

The rational approximation given by Jones [6] is

$$C(\kappa) = \frac{\frac{1}{2}\kappa^2 + c_1\kappa + c_2}{\kappa^2 + c_3\kappa + c_2} \tag{2.3.18}$$

with

$$\begin{aligned} c_1 &= -0.2808\iota \\ c_2 &= -0.01365 \\ c_3 &= -0.3455\iota. \end{aligned} \tag{2.3.19}$$

Consider Eq. 2.3.10. Substituting Eq. 2.3.14 and multiplying by $\kappa^2(\kappa^2 + c_3\kappa + c_2)$, we obtain

$$\begin{aligned} \Big(&(\kappa^4 + c_3\kappa^3 + c_2\kappa^2)\chi^2(\mathrm{M}_0 + \mathrm{G}_0) + (\kappa^3 + c_3\kappa^2 + c_2\kappa)\chi^2\mathrm{G}_1 \\ &+ \left(\frac{1}{2}\kappa^3 + c_1\kappa^2 + c_2\kappa\right)\chi^2\mathrm{G}_2 + \left(\frac{1}{2}\kappa^2 + c_1\kappa + c_2\right)\chi^2\mathrm{G}_3 \\ &- (\kappa^4 + c_3\kappa^3 + c_2\kappa^2)\chi\mathrm{D}_0 - (\kappa^4 + c_3\kappa^3 + c_2\kappa^2)\mathrm{K}_0 \Big)\mathbf{x} = \mathbf{0}. \end{aligned} \tag{2.3.20}$$

This simplifies to

$$(\kappa^4\chi^2\mathrm{A}_0 + \kappa^3\chi^2\mathrm{A}_1 + \kappa^2\chi^2\mathrm{A}_2 + \kappa\chi^2\mathrm{A}_3 + \chi^2\mathrm{A}_4 + \kappa^4\chi\mathrm{A}_5 + \kappa^3\chi\mathrm{A}_6 + \kappa^2\chi\mathrm{A}_7 + \kappa^4\mathrm{A}_8 + \kappa^3\mathrm{A}_9 + \kappa^2\mathrm{A}_{10})\mathbf{x} = \mathbf{0} \tag{2.3.21}$$

with

$$\begin{aligned}
\mathrm{A}_0 &= \mathrm{M}_0 + \mathrm{G}_0 \\
\mathrm{A}_1 &= c_3\mathrm{M}_0 + c_3\mathrm{G}_0 + \mathrm{G}_1 + \tfrac{1}{2}\mathrm{G}_2 \\
\mathrm{A}_2 &= c_2\mathrm{M}_0 + c_2\mathrm{G}_0 + c_3\mathrm{G}_1 + c_1\mathrm{G}_2 + \tfrac{1}{2}\mathrm{G}_3 \\
\mathrm{A}_3 &= c_2\mathrm{G}_1 + c_2\mathrm{G}_2 + c_1\mathrm{G}_3 \\
\mathrm{A}_4 &= c_2\mathrm{G}_3 \\
\mathrm{A}_5 &= -\mathrm{D}_0 \\
\mathrm{A}_6 &= -c_3\mathrm{D}_0 \\
\mathrm{A}_7 &= -c_2\mathrm{D}_0 \\
\mathrm{A}_8 &= -\mathrm{K}_0 \\
\mathrm{A}_9 &= -c_3\mathrm{K}_0 \\
\mathrm{A}_{10} &= -c_2\mathrm{K}_0
\end{aligned} \tag{2.3.22}$$

Eq. 2.3.21 is a fourth-order polynomial multiparameter eigenvalue problem. We develop a direct solver for this problem in Chapter 4.

The approximation to Theodorsen's function given by Swinney [7] is

$$C(\kappa) = \frac{1 + F(i\kappa)^\beta}{1 + 2F(i\kappa)^\beta} \tag{2.3.23}$$

with

$$\begin{aligned}
\beta &= \frac{5}{6} \\
F &= 2.19.
\end{aligned} \tag{2.3.24}$$

Substituting Eq. 2.3.23 into Eq. 2.3.10, multiplying by $\kappa^2\left(1 + 2F(i\kappa)^\beta\right)$, and expanding $(i\kappa)^\beta$ we obtain

$$\begin{aligned}
\Big(\Big(&\kappa^2\left(1 + 2\imath^\beta F\kappa^\beta\right)(\mathrm{M}_0 + \mathrm{G}_0) + \kappa\left(1 + 2\imath^\beta F\kappa^\beta\right)\mathrm{G}_1 + \kappa\mathrm{G}_2\left(1 + \imath^\beta F\kappa^\beta\right) \\
&+ \mathrm{G}_3\left(1 + \imath^\beta F\kappa^\beta\right)\Big)\chi^2 - \kappa^2\left(1 + 2\imath^\beta F\kappa^\beta\right)\mathrm{D}_0\chi \\
&- \kappa^2\left(1 + 2\imath^\beta F\kappa^\beta\right)\mathrm{K}_0\Big)\mathbf{x} = \mathbf{0}
\end{aligned} \tag{2.3.25}$$

or

$$\Big(\big(\kappa^2(\mathrm{M}_0+\mathrm{G}_0)+\kappa(\mathrm{G}_1+\mathrm{G}_2)+\mathrm{G}_3+k^{\beta+2}2\iota^\beta F(\mathrm{M}_0+\mathrm{G}_0)$$
$$+k^{\beta+1}\iota^\beta F(2\mathrm{G}_1+\mathrm{G}_2)+k^\beta\iota^\beta F\mathrm{G}_3\big)\chi^2-\kappa^2\mathrm{D}_0\chi \qquad (2.3.26)$$
$$-k^{\beta+2}2\iota^\beta F\mathrm{D}_0\chi-\kappa^2\mathrm{K}_0-k^{\beta+2}2\iota^\beta F\mathrm{K}_0\Big)\mathbf{x}=\mathbf{0}.$$

To make further progress we specify $\beta = 5/6$, which yields:

$$\Big(\big(\kappa^2(\mathrm{M}_0+\mathrm{G}_0)+\kappa(\mathrm{G}_1+\mathrm{G}_2)+\mathrm{G}_3+k^{\frac{17}{6}}2\iota^\beta F(\mathrm{M}_0+\mathrm{G}_0)$$
$$+k^{\frac{11}{6}}\iota^\beta F(2\mathrm{G}_1+\mathrm{G}_2)+k^{\frac{5}{6}}\iota^\beta F\mathrm{G}_3\Big)\chi^2-\kappa^2\mathrm{D}_0\chi-k^{\frac{17}{6}}2\iota^\beta F\mathrm{D}_0\chi \qquad (2.3.27)$$
$$-\kappa^2\mathrm{K}_0-k^{\frac{17}{6}}2\iota^\beta F\mathrm{K}_0\Big)\mathbf{x}=\mathbf{0}.$$

Let us then define a new eigenvalue variable $\hat{\kappa}=\sqrt[6]{\kappa}$, in which case the system becomes

$$\Big(\big(\hat{\kappa}^{12}\chi^2(\mathrm{M}_0+\mathrm{G}_0)+\hat{\kappa}^6\chi^2(\mathrm{G}_1+\mathrm{G}_2)+\mathrm{G}_3\chi^2+\hat{\kappa}^{17}\chi^2 2\iota^\beta F(\mathrm{M}_0+\mathrm{G}_0)$$
$$+\hat{\kappa}^{11}\chi^2\iota^\beta F(2\mathrm{G}_1+\mathrm{G}_2)+\hat{\kappa}^5\chi^2\iota^\beta F\mathrm{G}_3\big)-\hat{\kappa}^{12}\chi\mathrm{D}_0 \qquad (2.3.28)$$
$$-\hat{\kappa}^{17}\chi 2\iota^\beta F\mathrm{D}_0-\hat{\kappa}^{17}2\iota^\beta F\mathrm{K}_0-\hat{\kappa}^{12}\mathrm{K}_0\Big)\mathbf{x}=\mathbf{0}$$

and simplifies to

$$(\hat{\kappa}^{17}\chi^2\mathrm{A}_0+\hat{\kappa}^{12}\chi^2\mathrm{A}_1+\hat{\kappa}^{11}\chi^2\mathrm{A}_2+\hat{\kappa}^6\chi^2\mathrm{A}_3+\hat{\kappa}^5\chi^2\mathrm{A}_4+\chi^2\mathrm{A}_5+\hat{\kappa}^{17}\chi\mathrm{A}_6$$
$$+\hat{\kappa}^{12}\chi\mathrm{A}_7+\hat{\kappa}^{17}\mathrm{A}_8+\hat{\kappa}^{12}\mathrm{A}_9)\mathbf{x}=\mathbf{0} \qquad (2.3.29)$$

with

$$\begin{aligned}
\mathrm{A}_0&=2\iota^\beta F(\mathrm{M}_0+\mathrm{G}_0)\\
\mathrm{A}_1&=\mathrm{M}_0+\mathrm{G}_0\\
\mathrm{A}_2&=\iota^\beta F(2\mathrm{G}_1+\mathrm{G}_2)\\
\mathrm{A}_3&=\mathrm{G}_1+\mathrm{G}_2\\
\mathrm{A}_4&=\iota^\beta F\mathrm{G}_3\\
\mathrm{A}_5&=\mathrm{G}_3\\
\mathrm{A}_6&=-2\iota^\beta F\mathrm{D}_0\\
\mathrm{A}_7&=-\mathrm{D}_0\\
\mathrm{A}_8&=-2\iota^\beta F\mathrm{K}_0\\
\mathrm{A}_9&=-\mathrm{K}_0.
\end{aligned} \qquad (2.3.30)$$

Eq. 2.3.29 is a polynomial multi-parameter eigenvalue problem of degree 17 in $\hat{\kappa}$ and degree 2 in χ. We develop a direct solver for this problem in Chapter 4.

2.3.6 Parameter values

In our derivation so far we have not yet specified the values of the structural and aerodynamic parameters in Eq. 2.3.11. We devise this model to be as close as possible to the introductory model (Section 2.2) but of course with damping and unsteady effects still included. Table 2.3 presents the extra model parameters (dimensional and nondimensional) not specified in Tables 2.1 and 2.2. Note that, even though Theodorsen's aerodynamic model is significantly more complex than the original steady model, no further aerodynamic parameters need be specified.

Table 2.3: Extra parameters for damped models

Parameter	Value
bending damp. coeff. – d_h	1 Ns/m
torsional damp. coeff. – d_θ	1 Nms/rad
bending damp. ratio – ζ_h	1.4105 %
torsional damp. ratio – ζ_θ	2.3508 %

2.4 REFERENCES

[1] Hodges, D. H., and Pierce, G. A., 2011, Introduction to Structural Dynamics and Aeroelasticity, Cambridge University Press, New York, New York State, USA.

[2] Abramowitz, M., and Stegun, I., 1972, Handbook of Mathematical Functions, National Bureau of Standards, Washington D.C., USA.

[3] White, F. M., 2009, Fluid mechanics, McGraw-Hill, New York, New York State, USA.

[4] Bisplinghoff, R. L., Ashley, H., and Halfman, R. L., 1957, Aeroelasticity, Addison-Wesley, Reading, Massachusetts, USA.

[5] Brunton, S. L., and Rowley, C. W., 2013, "Empirical state-space representations for Theodorsen's lift model," Journal of Fluids and Structures, **38**, pp. 174–186.

[6] Jones, R., 1938, Operational treatment of the nonuniform-lift theory in airplane dynamics, National Advisory Committee for Aeronautics, Washington D.C., USA.

[7] Swinney, D., 1989, "A fractional calculus model of aeroelasticity," Master's Thesis, Air Force Institute of Technology.

Chapter 3
Visualisation methods

3.1 THE PURPOSE OF VISUALISATION METHODS

Visualisation methods allow us to visualise flutter points – their locations, relation to one another, nature and cause. This is in contrast to many direct and iterative solution methods, which supply the locations of the flutter points but nothing more. Visualisation methods are inevitably based on some form of enumeration procedure – solving a simple problem or evaluating a function over a grid of points. The results from this enumeration are then plotted, and the flutter points typically manifest themselves as some form of intersection. Visualisation methods can of course be used to provide reasonable numerical estimates of the location of the flutter points – for example by an interpolative procedure – but their main benefit is in providing contextual information. They provide information about the subcritical and supercritical dynamics of the system – below and above the flutter point, respectively. This information can be used to deduce the mechanisms leading to the creation of a flutter point, and these mechanisms may in turn inform aircraft design decisions.

In this chapter we will outline two visualisation methods which may be familiar to the reader: the modal damping and root locus plots. We will then consider visualisation methods based on multiparameter considerations. The reason we consider visualisation methods first out of all our methods (they would perhaps form a more logical progression if located after Chapter 6's work on unstructured solvers) is because we will need them to verify and interpret the other algorithms that we will develop later. We will make heavy use of the contour plot (Section 3.3), a new method for flutter point computation which is perhaps the most important development of this chapter. In Section 3.4 we also provide visual overviews of most of the systems that we described in Chapter 2; this will set the context for our analysis of these systems with different methods in the forthcoming chapters.

3.2 MODAL DAMPING AND ROOT LOCUS

The modal damping and root locus methods are widely used to analyse stability problems. These methods do not need to be understood in the context of multiparameter eigenvalue problems. Consider a general eigenvalue problem, representing an aeroelastic system:

$$\mathrm{A}(\chi, p)\mathbf{x} = 0. \tag{3.2.1}$$

The eigenvalue parameter is $\chi \in \mathbb{C}$, representing the modal frequency of the structure, and $p \in \mathbb{R}$ is another parameter which is some function of airspeed and modal frequency (e.g. airspeed U itself, or the reduced frequency $b\chi/U$). We define a set of p-values $\{p_k\}$ that are of interest, and at these values we solve the one-parameter eigenvalue problem for χ. The result is a set of solutions $\{\chi_i\}_k$, for each element in $\{p_k\}$. Note that, even if the dependency of A on χ is too complex for standard solvers, the solution can always be determined with further enumeration: for each p_k, we select a set of test points in complex space, $\{\chi_j^*\} \subset \mathbb{C}$ at which we evaluate $\det(A)$. We then interpolate the resulting determinant field to χ such that $\det\big(A(\chi, p_k)\big) = 0$. These points are the eigenvalues $\{\chi_i\}_k$. However, it should also be noted that this process is extremely expensive, as it adds an extra two dimensions ($\mathrm{Re}(\chi)$ and $\mathrm{Im}(\chi)$) to the normal one-dimensional search over p.

In terms of our traditional aeroelastic 'method' classifications, this procedure is most directly related to the p-method and the pk-method – the difference between these two methods is essentially one of system structure and not of solution method. The root locus and modal damping plots are essentially only different forms of visualisation for the information from this procedure. The modal damping plot shows the imaginary part of each eigenvalue (the modal damping) against the set of p-values; that is $\mathrm{Im}(\{\chi_i\}_k)$ against p_k. Where the modal damping of any mode becomes zero ($\mathrm{Im}(\{\chi_i\}_k) = 0$) flutter occurs. The modal damping plot is often complemented by a second plot showing the frequency of each mode against the set of p-values; $\mathrm{Re}(\{\chi_i\}_k)$ against p_k. The use of two separate plots can cause inconvenience, because it is often not possible to link a given model damping curve with its flutter curve via the plot alone. One has to investigate the raw data to determine this. For example, note that in Figure 3.1 it is not immediately obvious which of the branches in each of the plots are associated with one another. Modal damping plots are nevertheless extremely common in aeroelasticity and may be found in a wide variety of

applications [1–3]. Figure 3.1 shows an example of modal damping plot and corresponding frequency plot.

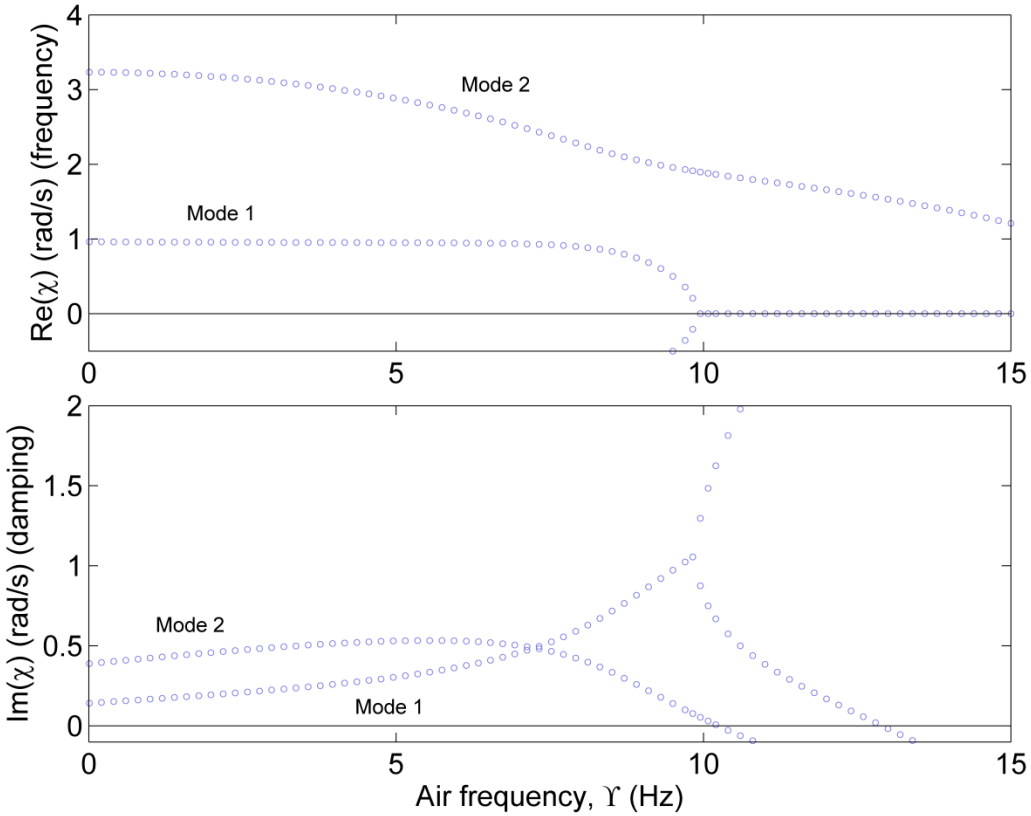


Figure 3.1: An example modal damping plot

The root locus method plots $\mathrm{Im}(\{\chi_i\}_k)$ against $\mathrm{Re}(\{\chi_i\}_k)$ for each eigenvalue. This has the advantage of fitting on a single plot, but it gives no direct graphical information about the dependency of the eigenvalues on the parameter p. This information has to be determined by looking at the raw numerical data. Figure 3.2 shows an example root-locus plot. Note that in our root locus plots, the lower-half plane is the unstable half-plane. This is due to the definition of our eigenvalue (cf. Ch. 1 Sect. 1.1). In most aeroelastic applications, the modal damping plot is more widespread than the root locus plot, as the latter does not give any indication of the location of the flutter points in with respect to the parameter p. However, it is still used [4,5].

As both root locus and modal damping plots are based around the same set of data, and so they share a common deficiency. They essentially rely on the fact that the dependence of Eq. 3.2.1 on χ is in most cases sufficiently simple that the system can be solved by a

standard (usually polynomial) eigenvalue solver. We did note that the use of a further enumeration / interpolation procedure be used for a more general system, but this procedure is seldom feasible due to the high computational expense.

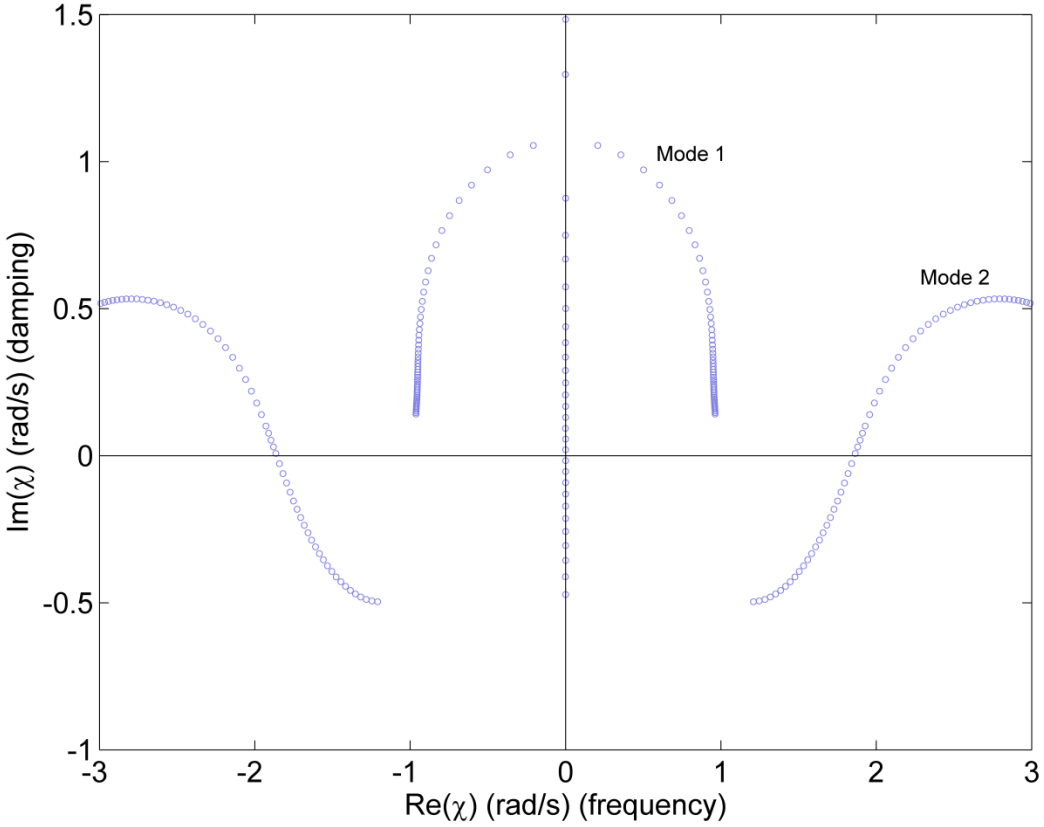


Figure 3.2: An example root locus plot

If we look back at the multiparameter formulation of our flutter problem, Eq. 3.2.1, where $[\chi, p] \in \mathbb{R}^2$, we can see that the essential idea behind the root locus and modal damping methods is the introduction of a complex component in the modal frequency χ. This complex component has physical meaning as the modal damping of an associated modal frequency. However, in our original system, there was no particular mathematical reason to introduce a complex component into χ as opposed to the other parameter (e.g. airspeed). We can thus devise a *reverse* enumeration, in which we define a set of χ-values $\{\chi_k\} \in \mathbb{R}$ that are of interest, and solve the one-parameter system at each of the values for $p \in \mathbb{C}$. The reverse modal damping plot shows the imaginary component of p against χ_k, and the reverse root locus plot shows the imaginary component of p against the real component of p. Figure 3.3 shows a reverse root locus plot, and Figure 3.4 a reverse modal damping plot, both for $p = \Upsilon$ (the air frequency). These reverse methods are put at a general disadvantage

to the standard methods by the fact that the complex components of p – whether p is airspeed, reduced frequency, or some other parameter – are more difficult to interpret physically than the complex components of χ. However, there are some areas in which the reverse methods may be useful. Firstly, they do overcome the deficiency of the standard modal damping / root locus methods that was raised earlier. They work for an arbitrary continuous dependence of Eq. 3.2.1 on χ. But they replace this with another deficiency: they now rely on this equation having a sufficiently simple dependence on p, because this is now the variable being solved for at each given χ. Unfortunately, the dependence of Eq. 3.2.1 on p (no matter what airspeed parameter p represents) is usually more complex that its dependence on χ, so that the standard methods will usually be more practical than the reverse ones. However, there are some cases where the reverse methods may be practical. For example, the flutter analysis of long-span bridges is often carried out with experimental aerodynamic models base on flutter derivative parameters [6–8]. If these flutter derivatives are not allowed to vary with airspeed, then the Eq. 3.2.1 becomes quadratic with respect to airspeed. If this is so, then the reverse root locus methods allow for an arbitrary continuous dependence of Eq. 3.2.1 – for example, viscoelastic effects could be included in the problem with essentially no added computational time. However, it is questionable whether a model based on constant flutter derivatives would be accurate enough to be industrially relevant, as these derivatives usually vary [6–8].

If a system with sufficiently simple dependence on p is found, then the reverse methods offer quite a different perspective into the behaviour of the system than the standard ones. This is simply a result of the nature of the initial set of points of interest, $\{p_k\}$ or $\{\chi_k\}$. The standard methods will compute all the flutter points (at any mode) over a given airspeed range (the range of $\{p_k\}$). The reverse methods will compute all the flutter points over the given frequency range (the range of $\{\chi_k\}$). Figure 3.5 indicates this difference, on a plot of $\chi \in \mathbb{R}$ against $p \in \mathbb{R}$. The standard methods are useful when there is a small range of p-values of interest, whereas the reverse methods are useful when there is a small range of frequencies of interest. Note that this consideration does not apply when an enumeration / interpolation procedure is being used to solve the eigenvalue problem at any of the points of interest. In this case, one can specify the range of both variables.

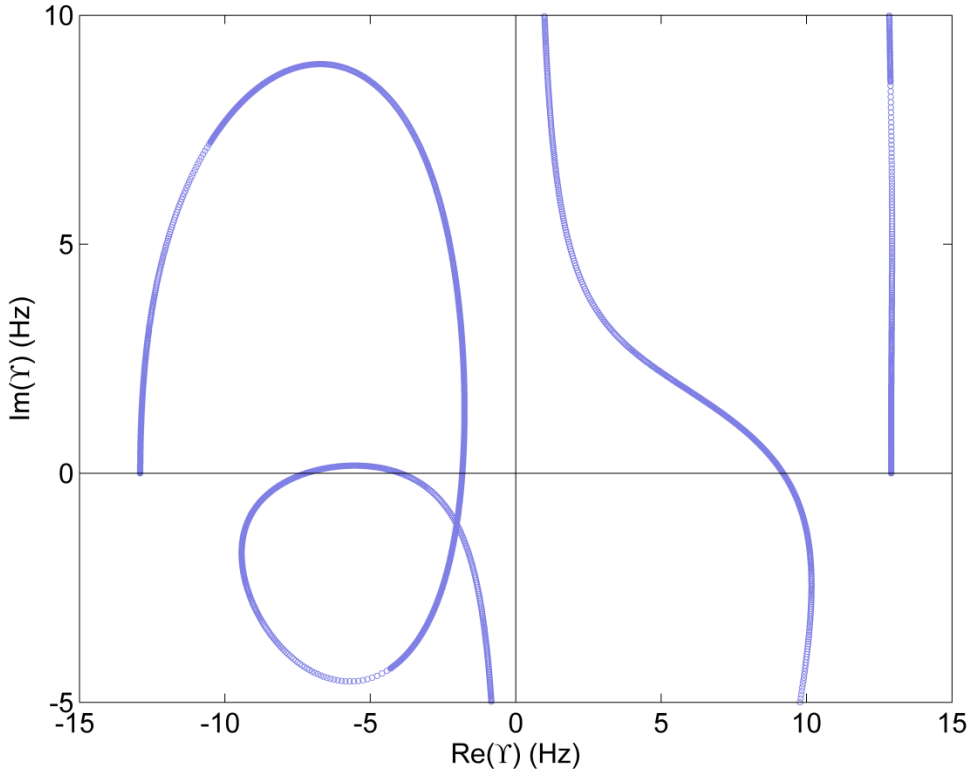


Figure 3.3: An example reverse root locus plot

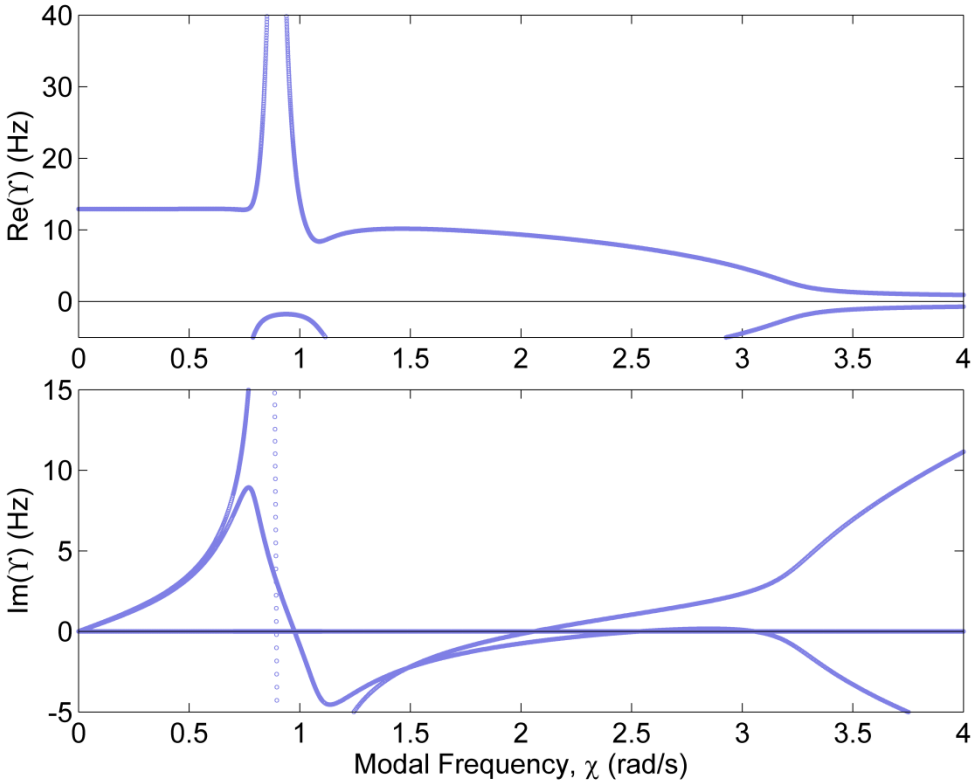


Figure 3.4: An example reverse modal damping plot. Note the areas where a higher $\{\chi_k\}$ resolution has been employed to resolve the steep gradients.

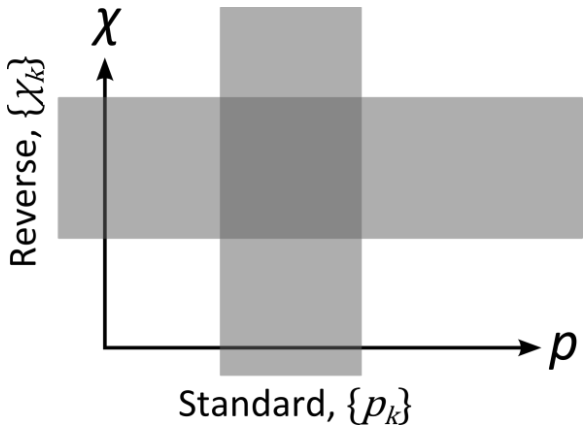


Figure 3.5: Solution zones for standard and reverse modal damping plots.

One thing that should be noted about the reverse methods is that it is impossible to distinguish a destabilisation event (modal damping changing from positive to negative) from a restabilisation even (modal damping changing from negative to positive). While it may be possible to do so, the author has not yet found a relationship between the imaginary p and the modal damping that would allow one to make such a distinction. Hence our reverse methods will only locate points on the flutter boundary, and will not describe their nature. However in standard aeroelastic analysis this is not a problem for two reasons. Firstly, because in a physical aeroelastic system we should expect the restabilisation to always occur at higher airspeeds than the destabilisation. Making this assumption, it is easy to distinguish these two points (along any given modal line, we will have a series of alternating restabilisations and destabilisations). Secondly, because the restabilisation points are of no known industrial relevance anyway. Aeroelastic engineers do not usually deal with structures that are unstable at zero airspeed and then stabilise once the aircraft is moving.

There are perhaps more potential applications of these reverse methods in areas other than aeroelasticity. There is particular potential for use in areas where the dependency of a system on its modal frequency may be quite complex, e.g. viscoelasticity, but the dependence p may be simple. In most industrial aeroelastic cases the standard methods will be preferable to the reverse ones – but aeroelasticians should be aware that a choice does exist.

3.3 CONTOUR PLOT

When we devised the reverse modal damping / root locus methods, we were essentially adapting existing methods to a multiparameter context. However, with some consideration we can devise a more powerful method which does not reference these existing modal damping / root locus techniques. Consider again the original eigenvalue problem:

$$\mathrm{A}(\chi, p)\mathbf{x} = 0. \tag{3.3.1}$$

If $\chi \in \mathbb{R}$ and $p \in \mathbb{R}$, then a solution only exists on the stability boundary, and nowhere else. However, $\det\big(\mathrm{A}(\chi, p)\big)$ does exist over $[\chi, p] \in \mathbb{R}^2$. The eigenvalues of the system will be the roots of the determinant:

$$\det\big(\mathrm{A}(\chi, p)\big) = 0. \tag{3.3.2}$$

In general, $\det\big(\mathrm{A}(\chi, p)\big) \in \mathbb{C}$. Hence Eq. 2.3 is equivalent to specifying

$$\begin{aligned} \mathrm{Re}\big(\det\big(\mathrm{A}(\chi, p)\big)\big) &= 0, \\ \mathrm{Im}\big(\det\big(\mathrm{A}(\chi, p)\big)\big) &= 0. \end{aligned} \tag{3.3.3}$$

Eq. 3.3.3 defines two sets of contours over $[\chi, p] \in \mathbb{R}^2$. The flutter points are the intersections of these two sets of contours. Dynamic flutter points are intersections that occur at nonzero χ, and divergence points are intersections occurring theoretically at $\chi = 0$; but due to the inevitable numerical errors in the interpolation of these intersections, they will appear at small negative or positive χ. The procedure for determining the flutter points is thus as follows:

1. Define a grid of points $\{[\chi_k, p_k]\} \subset \mathbb{R}^2$
2. Evaluate the complex $d_k = \det\big(\mathrm{A}(\chi_k, p_k)\big)$
3. Plot the contours of $\mathrm{Re}(d_k) = 0$ and $\mathrm{Im}(d_k) = 0$.

Figure 3.6 shows an example of a contour plot, for a Kirchhoff plate model of a biplane wing [9]. Note the divergence point at slightly past 20 m/s, and the flutter point at slightly below 100 m/s.

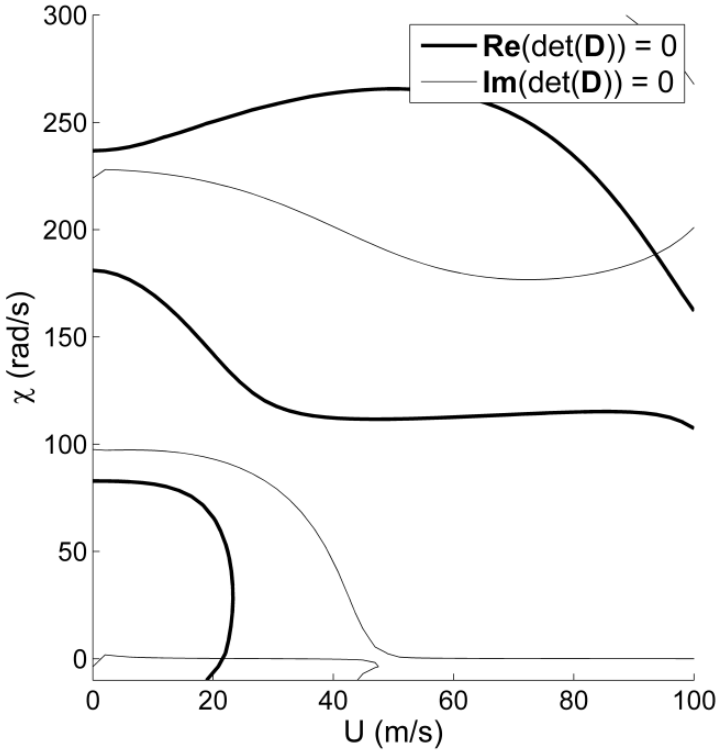


Figure 3.6: An example contour plot

The most significant advantage of this method is its applicability to unstructured systems – that is, for which neither variable can be solved with a direct solver. We saw earlier that the modal damping / root locus methods are capable of handling such system, but at the expense of requiring three enumerations in total: the initial $\{p_k\} \in \mathbb{R}$, and then $\{\chi_k\} \in \mathbb{C}$ (that is, $\{\mathrm{Re}(\chi_k)\} \in \mathbb{R}$ and $\{\mathrm{Im}(\chi_k)\} \in \mathbb{R}$). The contour plot requires only two enumerations: $\{p_k\} \in \mathbb{R}$ and $\{\chi_k\} \in \mathbb{R}$. We have removed a whole dimension from the search space.

The contour plot method has other advantages, as well. Unlike the modal damping methods (reverse or standard), it is immediately obvious which flutter speeds are associated with which flutter frequencies. Also, the real contours, $\mathrm{Re}\big(\det\big(\mathrm{A}(\chi, p)\big)\big) = 0$, approximate the natural frequencies of the structure as a function of airspeed. The relationship here is complex – these lines represent neither the damped nor the undamped natural frequencies of the structure. The difference between the damped natural frequencies and the real contours in relatively easy to grasp: the damped natural frequencies represent the real part of the complex frequency such that $\det(\mathrm{A}) = 0$, which is usually going to be different from the real frequency at which $\mathrm{Re}(\det(\mathrm{A})) = 0$. However, the difference between the

undamped natural frequencies and the real contours is more subtle. These undamped natural frequencies are the lines of $\det\big(\mathrm{Re}(\mathrm{A})\big) = 0$, whereas the real contours are $\mathrm{Re}(\det(\mathrm{A})) = 0$. The difference is that in general $\det\big(\mathrm{Re}(\mathrm{A})\big) \neq \mathrm{Re}(\det(\mathrm{A}))$ for complex A. At the flutter points, the damped natural frequencies are equivalent to the real contours. This is self-evident: at the flutter points $\chi \in \mathbb{R}$ and $\det(\mathrm{A}) = 0$ so the two definitions merge. The undamped natural frequencies are *not* equivalent, because the fact that $\det(\mathrm{A}) = 0$ does not imply $\det\big(\mathrm{Re}(\mathrm{A})\big) = 0$ for complex A. Of course, if the system is entirely undamped[1] then the damped and undamped natural frequencies coalesce, and we can easily show that the real contours are equivalent to this coalesced natural frequency. Firstly, we define the system $\mathrm{A}(\chi, p)$ as being undamped if:

$$\forall[\chi, p] \in \mathbb{R}^2, \mathrm{A}(\chi, p) \in \mathbb{R}. \tag{3.3.4}$$

That is, A is never complex-valued for any real χ or p. In a polynomial matrices system this is equivalent to all the matrix coefficients being real. The natural frequencies of such a system (for given p) are the eigenvalues, $\chi : \det\big(\mathrm{A}(\chi, p)\big) = 0$. If Eq. 3.3.4 holds, then $\det\big(\mathrm{A}(\chi, p)\big) \in \mathbb{R}$ and hence $\mathrm{Re}\big(\det\big(\mathrm{A}(\chi, p)\big)\big) = \det\big(\mathrm{A}(\chi, p)\big)$. That is, the real contours are equivalent to the natural frequencies of the structure as a function of p.

While we have not proved any error bounds, numerical experimentation indicates that the real contours in the contour plots are good approximations of the natural frequencies of modes with low modal damping. We present some of these experiments in Section 3.4. This makes the contour plot method particularly attractive for visualising systems formulated using the pk-method: the pk-method only produces good approximations of the modal behaviour in the lightly-damped modes anyway.

[1] Note: not only structurally undamped, but also having no aerodynamic damping.

3.4 NUMERICAL EXPERIMENTS

3.4.1 Simple section model

To begin our numerical experiments we will consider the introductory model described in Chapter 2:

$$(\mathrm{A} + \mathrm{B}\chi^2 + \mathrm{C}\Upsilon^2)\mathbf{x} = 0. \tag{3.4.1}$$

The purpose of experimenting with this system is to validate the most basic methods that we will be using – the modal damping and root locus plots – and to show that our methods are capable of replicating established results. Ironically, this system is one of the most troublesome to analyse by the methods we will look at in this thesis, and serves to illustrate the complexity that even simple systems can pose.

Figure 3.6 shows a modal damping plot for this system. There are two aeroelastic events over the simulated airspeed range: a flutter point and then a divergence point. We locate the divergence point at approximately $\Upsilon_D = 4.0\ \mathrm{Hz}$ and the flutter point at $\Upsilon_F = 2.6\ \mathrm{Hz}$ and $\chi_F = 0.78\ \mathrm{rad/s}$. The divergence point agrees very well with the analytical solution obtained in [2], namely $\Upsilon_D = 3.990\ \mathrm{Hz}$. The flutter point matches well with the value obtained numerically in [2], which is $\Upsilon_F = 2.600\ \mathrm{Hz}$ and $\chi_F = 0.7853\ \mathrm{rad/s}$ (see Chapter 2). This validates our implementation of the modal damping method. We can then compare this solution to that of the other methods. Figure 3.7 shows a root locus plot for this system. The plot gives no direct graphical information as to the location of the flutter and divergence points in Υ, but the flutter frequency can be observed to be $\chi_F = 0.78\ \mathrm{rad/s}$. The problematic aspect of this system can be seen in Figure 3.6: the system has zero modal damping for all airspeeds apart from the gap at $2.600\ \mathrm{Hz} < \Upsilon < 3.990\ \mathrm{Hz}$. Every airspeed outside this gap is technically a flutter point, as it has a neutral stability. Hence the multiparameter problem of determining the flutter points of the system has a continuous spectrum. We can see this when we consider the contour plot of this system (Figure 3.8). Note that $\mathrm{Im}(d_k)$ is identically zero everywhere. Every point outside the range $2.60\ \mathrm{Hz} < \Upsilon < 3.99\ \mathrm{Hz}$ (where the real contours do not exist) is thus a flutter point. The continuous nature of this problem's spectrum will cause difficulties when we attempt to solve it with direct solvers, as we shall do in Chapter 4.

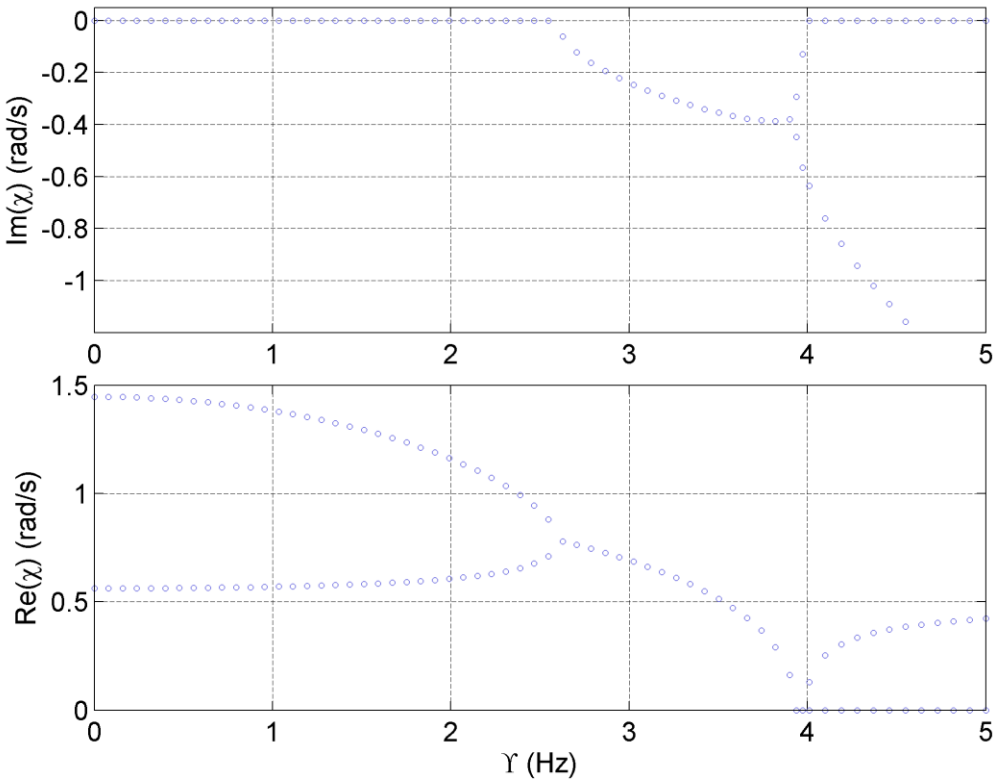


Figure 3.6: Modal damping plot for introductory system

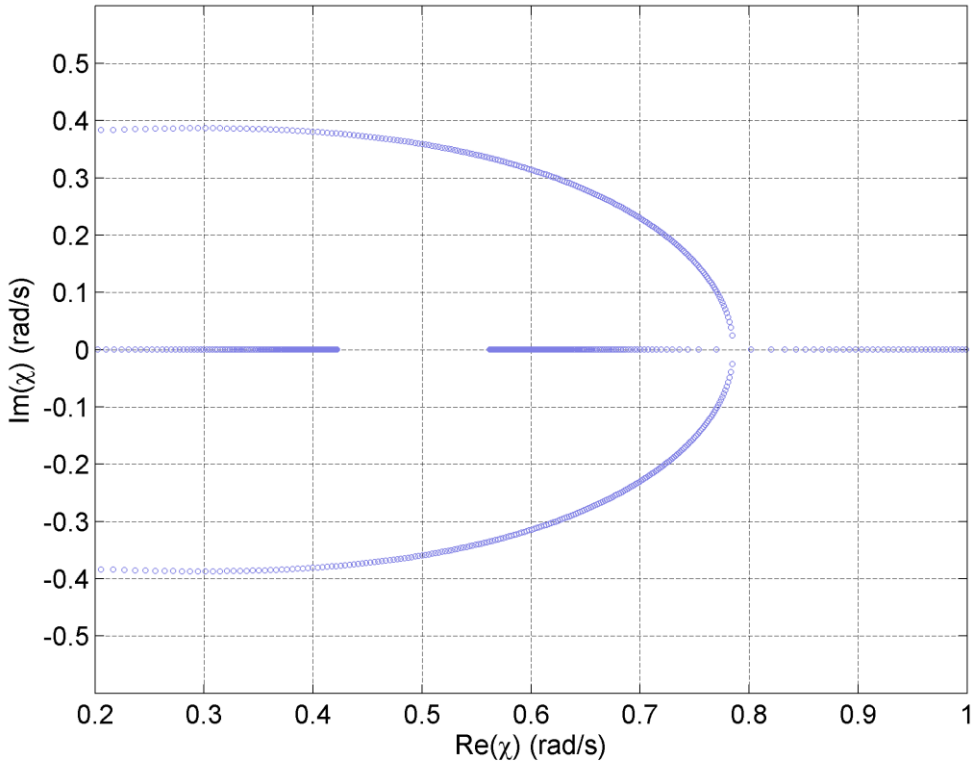


Figure 3.7: Root locus plot for introductory system

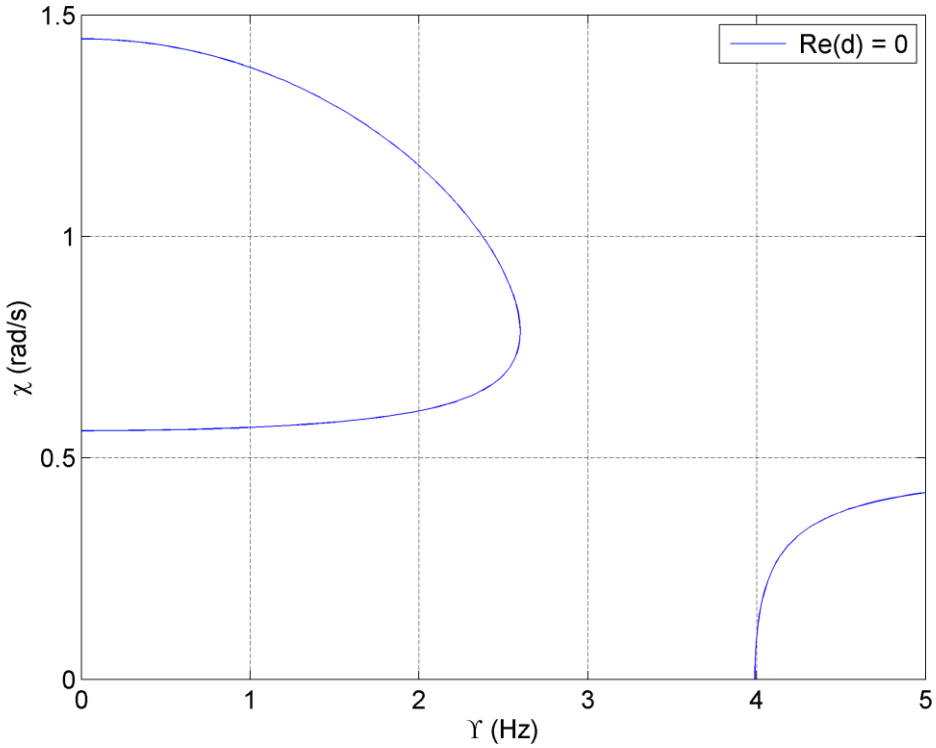


Figure 3.8: Contour plot of introductory system. The imaginary part of the determinant is zero everywhere.

3.4.2 Undamped section model with quasisteady aerodynamics

We now simulate a more complex model: the section model with quasisteady Theodorsen aerodynamics and no structural damping (Eq. 3.4.2).

$$\left((\mathrm{M}_0 + \mathrm{G}_0) + (\mathrm{G}_1 + \mathrm{G}_2)\tau + \mathrm{G}_3\tau^2 - \mathrm{K}_0\Lambda\right)\mathbf{x} = \mathbf{0}, \qquad (3.4.2)$$

Figure 3.9 shows a contour plot of this system. The plot range covers both positive and negative χ, so that its symmetry may be appreciated. According to this contour plot, the flutter point is located at $\tau_F = 0.996$, $\Lambda_F = 0.568\, s^2/\mathrm{rad}^2$. One non-physical flutter point can be located at $\tau_F = -0.996$, and two other points of neutral stability at $\tau = 0$ and $\Lambda = 3.33\, s^2/\mathrm{rad}^2$, $\Lambda = 0.492\, s^2/\mathrm{rad}^2$. The latter are physical but should not be regarded as a flutter point: it simply indicates that the system is undamped at zero airspeed ($\tau = 0$), which is exactly what would we expect, because the system has no structural damping and the aerodynamic damping $\mathrm{Im}(\mathrm{G}_1 + \mathrm{G}_2)\tau$ is proportional to τ (cf. Eq. 3.4.2). The divergence point of this system occurs at infinite τ and Λ and so cannot be located by the contour plot.

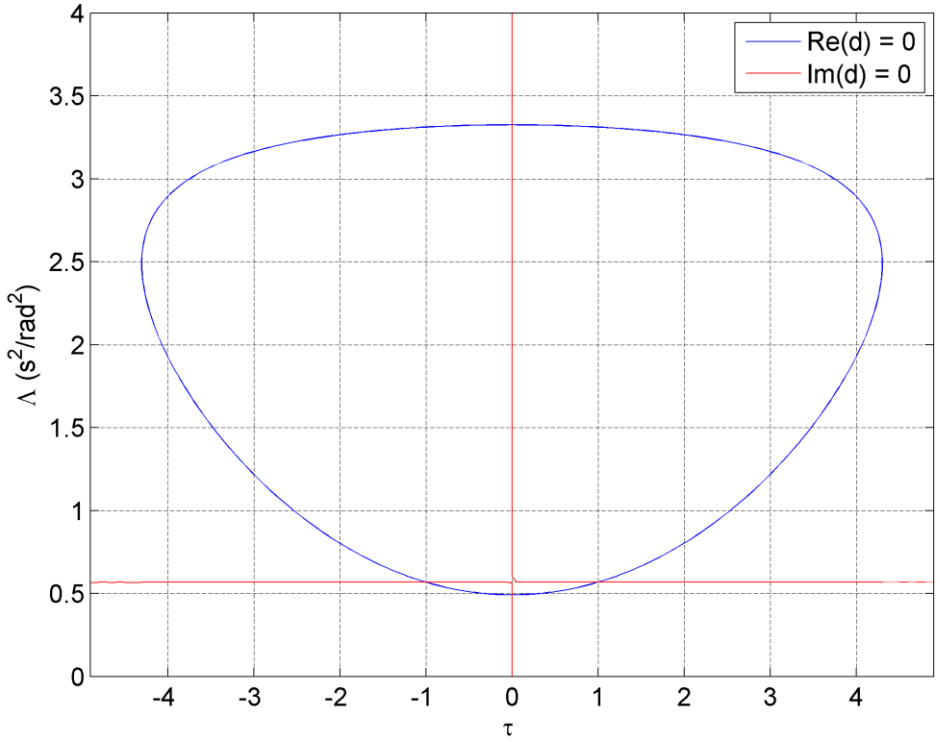


Figure 3.9: Contour plot for the section model with quasisteady Theodorsen aerodynamics and no structural damping

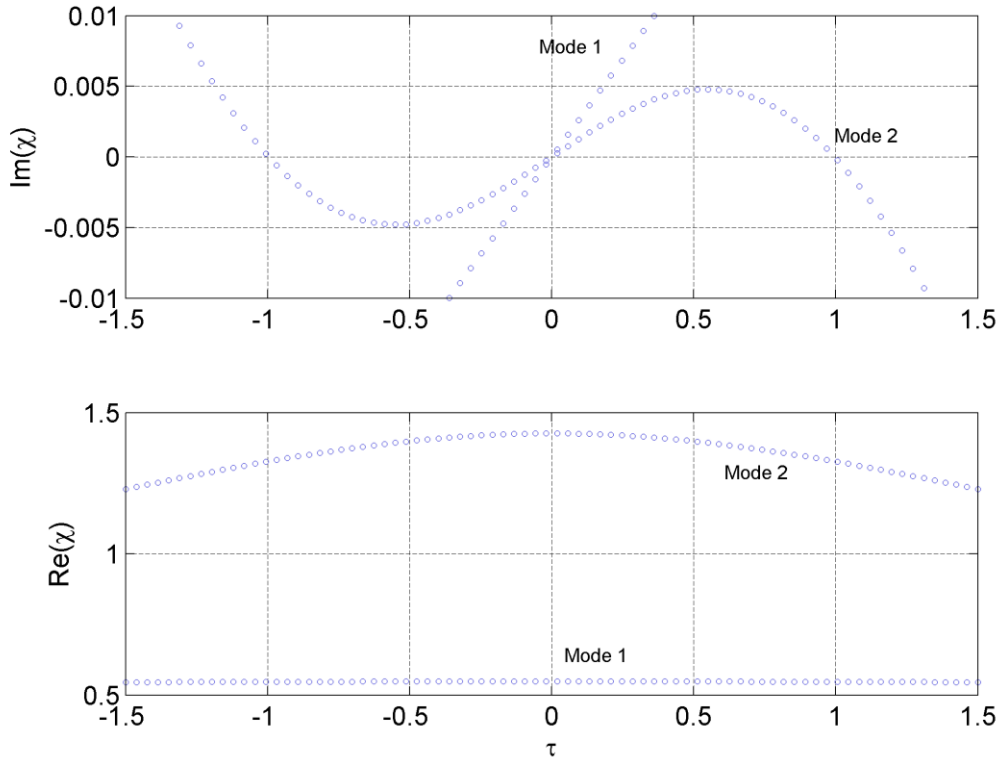


Figure 3.10: Modal damping plot for the section model with quasisteady Theodorsen aerodynamics and no structural damping

To validate this contour plot, we produce a modal damping plot. Figure 3.10 shows such a plot. Flutter can be seen to occur at $\tau_F = \pm 1.00$. We can link the modal damping curve for this flutter point with the lower modal frequency path in the $\mathrm{Re}(\chi)$ plot – though this is not obvious from the plot, and we have to investigate the raw data to determine this. We thus can estimate that $\chi_F = 1.32\ \mathrm{rad/s}$ ($\Lambda_F = 0.57\ s^2/\mathrm{rad}^2$). The two points of neutral stability lie at $\tau = 0$, $\chi = 0.548\ \mathrm{rad/s}$ and $\chi = 1.426\ \mathrm{rad/s}$ ($\Lambda = 3.33\ s^2/\mathrm{rad}^2$ and $\Lambda = 0.492\ s^2/\mathrm{rad}^2$). This agrees well with the contour plot.

3.4.3 Section model with quasisteady aerodynamics

As an extension of Section 3.4.2, we simulate the quasisteady section model with structural damping now included. This is a system which could be industrially useful to visualise. In Chapter 2 we presented two forms of this system: the form in τ-λ and the form in Υ-χ. In this section we solve both forms, and compare them to the undamped quasistatic section model (Section 2.4.2) and the introductory model (Section 2.4.1) respectively. Consider first the τ-λ form:

$$\left((\mathrm{M}_0 + \mathrm{G}_0) + (\mathrm{G}_1 + \mathrm{G}_2)\tau + \mathrm{G}_3\tau^2 - \mathrm{D}_0\lambda - \mathrm{K}_0\lambda^2\right)\mathbf{x} = \mathbf{0}, \tag{3.4.3}$$

Figure 3.11 shows the contour plot for the damped quasisteady system, superimposed on the contour plot for the undamped quasisteady system (mapped from Λ to λ). As can be seen, the real contours for these two plots are practically identical. The imaginary contours are similar in general trend, but the damped imaginary contours are now apparently hyperbolic in nature (though in fact they are quartic plane curves). Note that the symmetry of the plot is now broken. The physical flutter point in Figure 3.11 can be located at $\tau = 1.65$ and $\lambda = 0.834\ \mathrm{s/rad}$. Only one non-physical flutter point exists, at $\tau = -0.271$ and $\lambda = 1.82\ \mathrm{s/rad}$. The physical flutter point for the damped model is reasonably different to that of the undamped model, being over one-and-half times greater τ and 10% higher in λ. This indicates the difference that a small change in structural damping can effect on the aeroelastic behaviour of the wing structure.

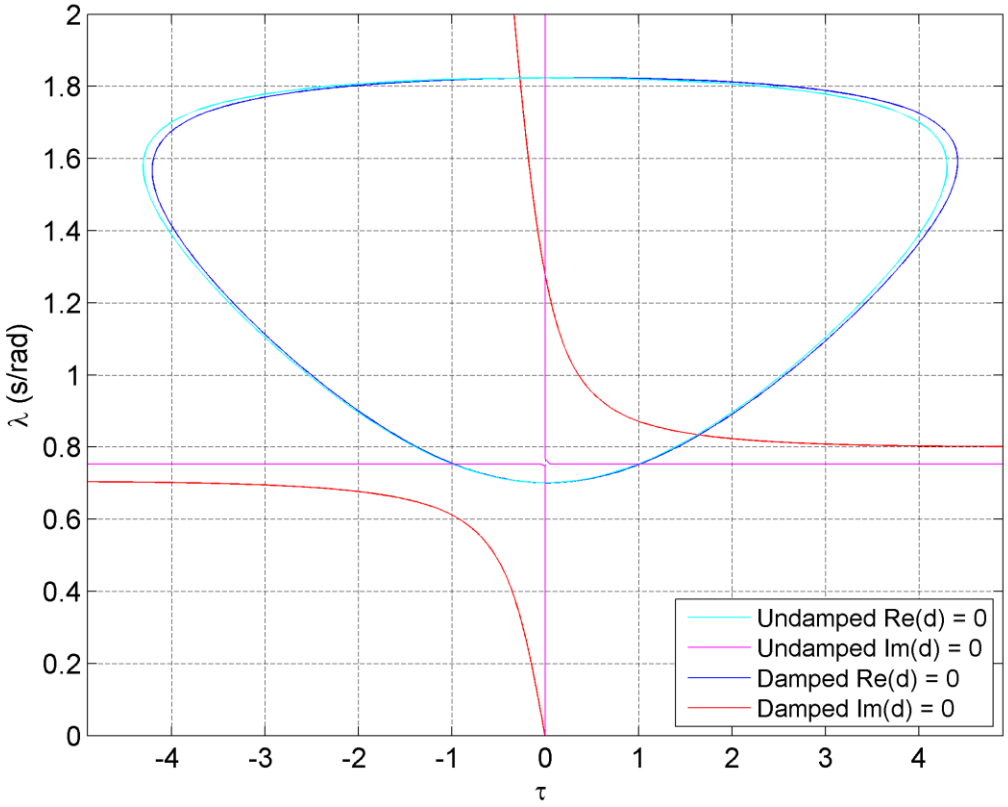


Figure 3.11: Comparison between the contour plots of the damped and undamped quasisteady system.

Consider now the Υ-χ form of the system:

$$\left((\mathrm{M}_0 + \mathrm{G}_0)\chi^2 + (\mathrm{G}_1 + \mathrm{G}_2)\Upsilon\chi + \mathrm{G}_3\Upsilon^2 - \mathrm{D}_0\chi - \mathrm{K}_0\right)\mathbf{x} = \mathbf{0}, \tag{3.4.4}$$

Figure 3.12 shows a modal damping plot for this system, superimposed on that of the steady introductory model (see Section 2.4.1). The flutter point for the damped model in Figure 3.12 can be located at $\Upsilon_F = 1.98\ \mathrm{Hz}$. From the raw data we can link the associated modal damping curve with the upper modal frequency path in the $\mathrm{Re}(\chi)$ plot, and hence we estimate that $\chi_F = 1.20\ \mathrm{rad/s}$. This corresponds to $\tau = 1.65$ and $\lambda = 0.833\ \mathrm{s/rad}$, agreeing with the results from the τ-λ form. The divergence point can be located at $\Upsilon_D = 3.99\ \mathrm{Hz}$, which matches the divergence point of the steady introductory model exactly. This is what we would expect, because the difference between the quasisteady Theodorsen and steady aerodynamic theories is their treatment of the dynamic motion of the wing – the steady theory ignores dynamic motion whereas the quasisteady Theodorsen

theory models it in a quasistatic way. Divergence does not involve any dynamic motion and so the two models agree. This gives credence to our formulation of Theodorsen's theory in Chapter 2, and verifies that our nondimensionalisation is working correctly.

It may be noted that the quasisteady section model has sufficiently simple dependence on Υ that we can use the reverse modal damping / root locus methods. Figure 3.13 shows a reverse modal damping plot for this system. We can locate the divergence point at $\Upsilon_D = 3.99$ Hz, and the flutter point at $\Upsilon_F = 1.98$ Hz and $\chi_F = 1.2$ rad/s. Figure 3.14 shows an equivalent reverse root locus plot. Note that the two loci that exit the plot in the top and bottom of the right half plane are actually one and the same – they are part of a massive circular locus that extends to the right of this plot. The same flutter points can be identified (though their χ-values cannot be directly determined) and these points agree well with our modal damping plot (Figure 3.12).

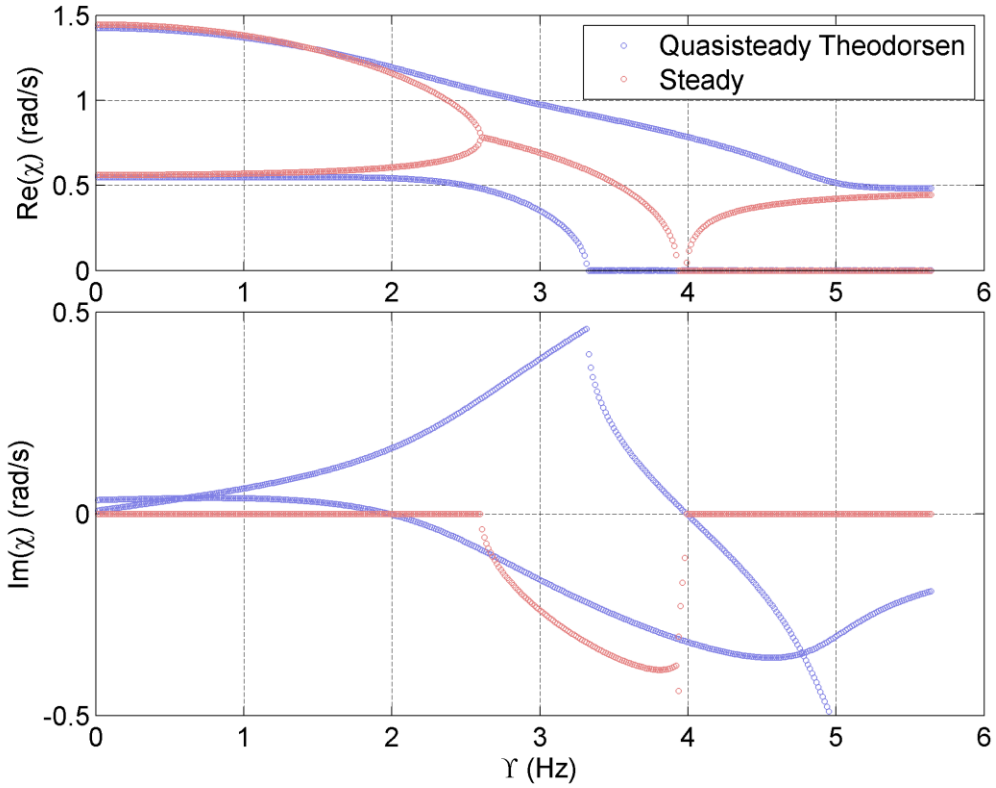


Figure 3.12: Comparison between modal damping plots for our section model with quasisteady aerodynamics and structural damping, and with steady aerodynamics.

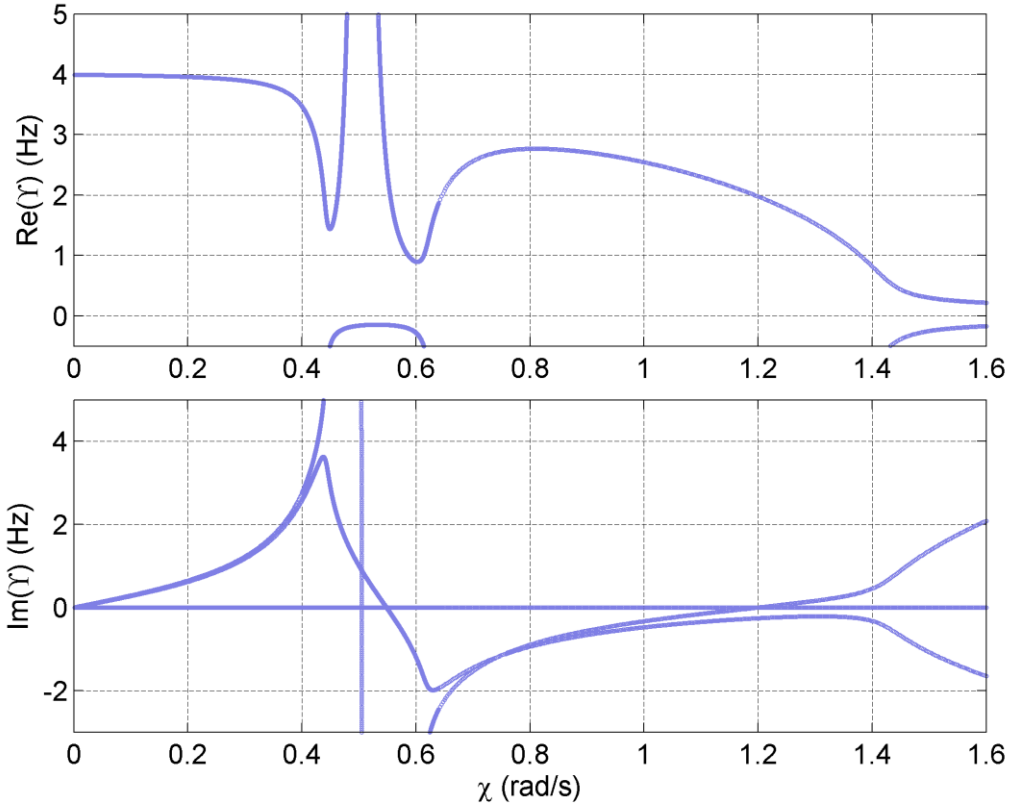


Figure 3.13: Reverse modal damping plot for Eq. 3.4.4

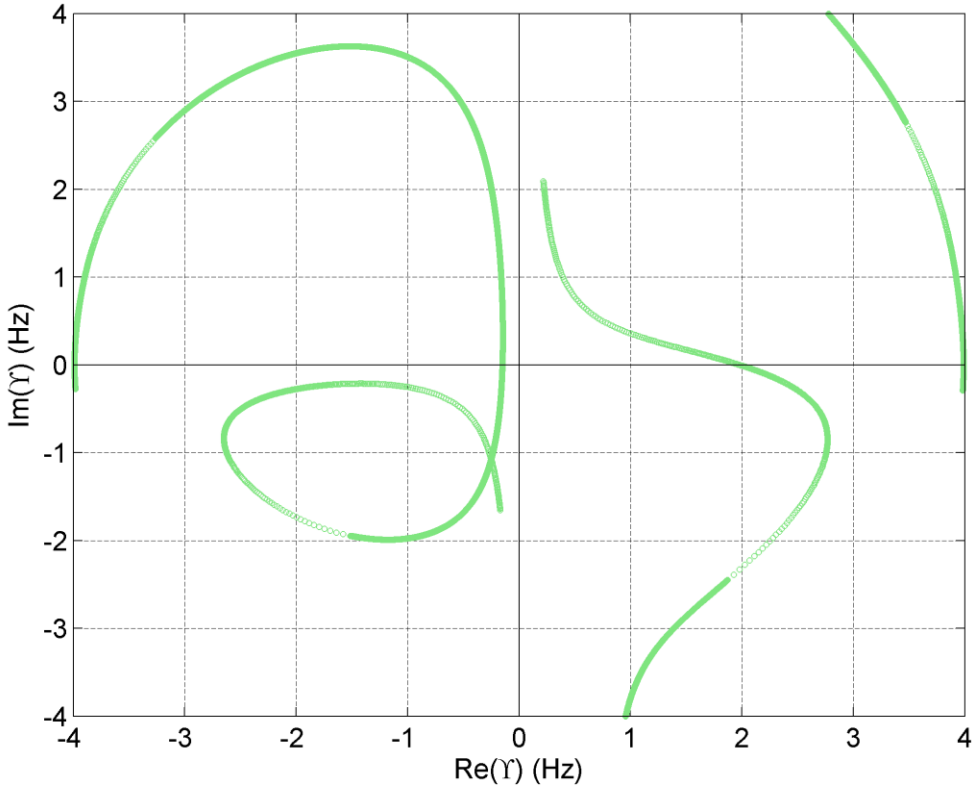


Figure 3.14: Reverse root locus plot for Eq. 3.4.4

However, two other points also show up on the plot. The first is a flutter point occurring at negative airspeed: this is at the point at $\chi_F = 0.548\ \mathrm{rad/s}$ and $\Upsilon_F = -0.149\ \mathrm{Hz}$. As we are defining our search χ, we can eliminate points at negative χ but not at negative airspeed. The other is a flutter point at extremely high airspeed: this occurs at $\chi = 0.50\ \mathrm{rad/s}$ and $\Upsilon = 93.7\ \mathrm{Hz}$. We cannot tell from the reverse modal damping plot whether this flutter point is a destabilisation or a restabilisation – given that this mode has gone through divergence at $\chi = 0$ and so it would be reasonable to expect this to be a restabilisation. However, we cannot actually prove this with any rigour, as there is nothing but physical intuition to prove that the divergence point is actually a destabilisation (as opposed to a restabilisation). However, we can resolve this ambiguity with a standard modal damping plot (Figure 3.15). As can be seen, the modal damping becomes positive at this event – it is a restabilisation of the divergent mode. The fact that the reverse methods pick up these other flutter points could be an advantage or a disadvantage, depending on the application. On the one hand, it is an efficient way of calculating the flutter points over a wide range of airspeeds. Alternately, it could be distracting to an engineer looking only for the conventional flutter points.

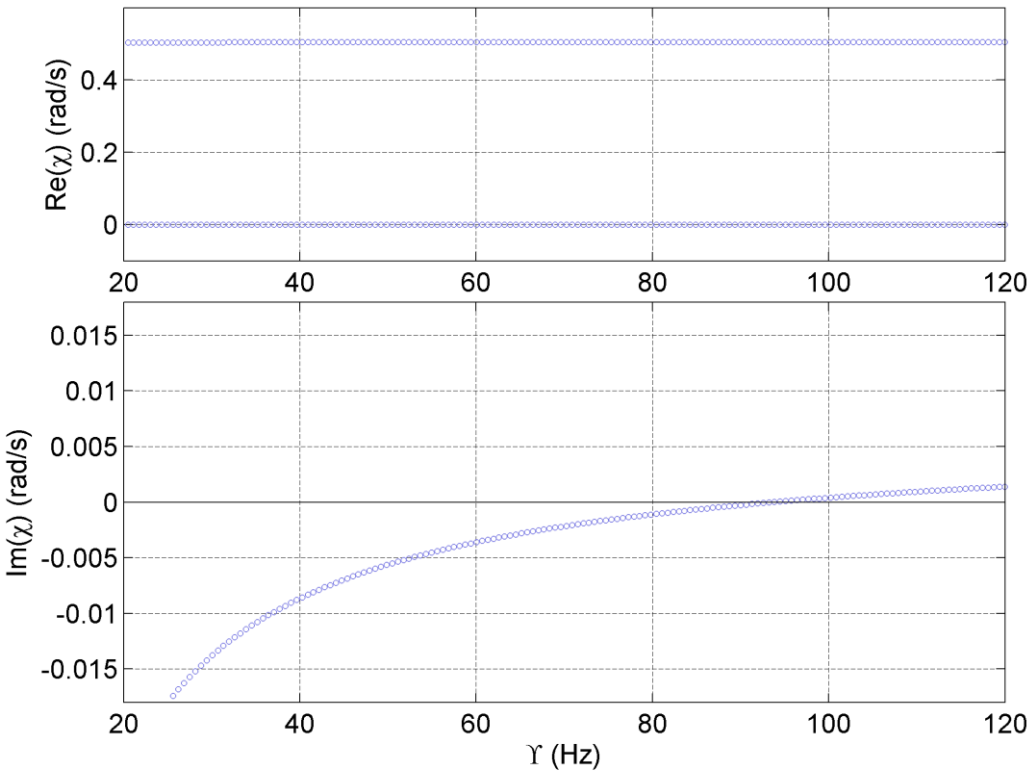


Figure 3.15: Modal damping plot for Eq. 3.4.4 showing restabilisation point

Lastly, we can produce a contour plot of Eq. 3.4.4. Figure 3.16 shows such a plot, superimposed on the damped and undamped natural frequency paths. The damped natural frequencies are the frequency paths shown in the standard modal damping plot (Figure 3.12), and the undamped natural frequencies are the eigenvalues (in χ) of the polynomial system:

$$\left(\mathrm{Re}(\mathrm{M}_0 + \mathrm{G}_0)\chi^2 + \mathrm{Re}\,(\mathrm{G}_1 + \mathrm{G}_2)\Upsilon\chi + \mathrm{Re}(\mathrm{G}_3)\Upsilon^2 - \mathrm{Re}(\mathrm{D}_0)\chi - \mathrm{Re}(\mathrm{K}_0)\right)\mathbf{x} = \mathbf{0}, \qquad (3.4.5)$$

as a function of Υ. From Figure 3.16 we can again locate the flutter point at $\Upsilon_F = 1.98\ \mathrm{Hz}$ and $\chi_F = 1.20\ \mathrm{rad/s}$, and the divergence point at $\Upsilon_F = 3.99\ \mathrm{Hz}$. At the flutter points, the real contour matches the damped natural frequency path but not the undamped natural frequency path. The real contours approximate the damped natural frequency paths well in areas of low damping (at low and high airspeeds). In the middle, between the flutter point and the divergence point, there is an area of high damping and so none of the three frequency paths agree well. While it may seem strange that damping increases nearer to the divergence point, recall Figure 3.12: there is usually a sharp damping peak followed by a sudden plunge. If we widen the field of view in Υ then we can see the other flutter points that were noted in the reverse modal damping plot. Figure 3.17 shows a wider contour plot. The restabilisation point can be seen on the far right, the negative flutter point slightly below $\Upsilon = 0$, and another divergence point at negative airspeed. This other divergence point is present (but outside the field of view) in all our previous plots.

Although the modal damping plot gives the most complete information about the system (being the only method that gives an exact measure of modal damping at subcritical and supercritical airspeeds), the other methods are far preferable in certain situations. The reverse methods are useful when the system's dependence on the airspeed parameter is simple. The contour plot is much less computationally intensive than any of other methods when used on unstructured systems. The contour plot is also the best for getting a comprehensive picture of the behaviour of the system: it is the only method which allows one to compare the locations of the flutter points in both χ and p from the plot alone. For these reasons we will be using the contour plot extensively to verify the algorithms we will develop in the following chapters, as we will often be dealing with systems which are semi-structured or unstructured.

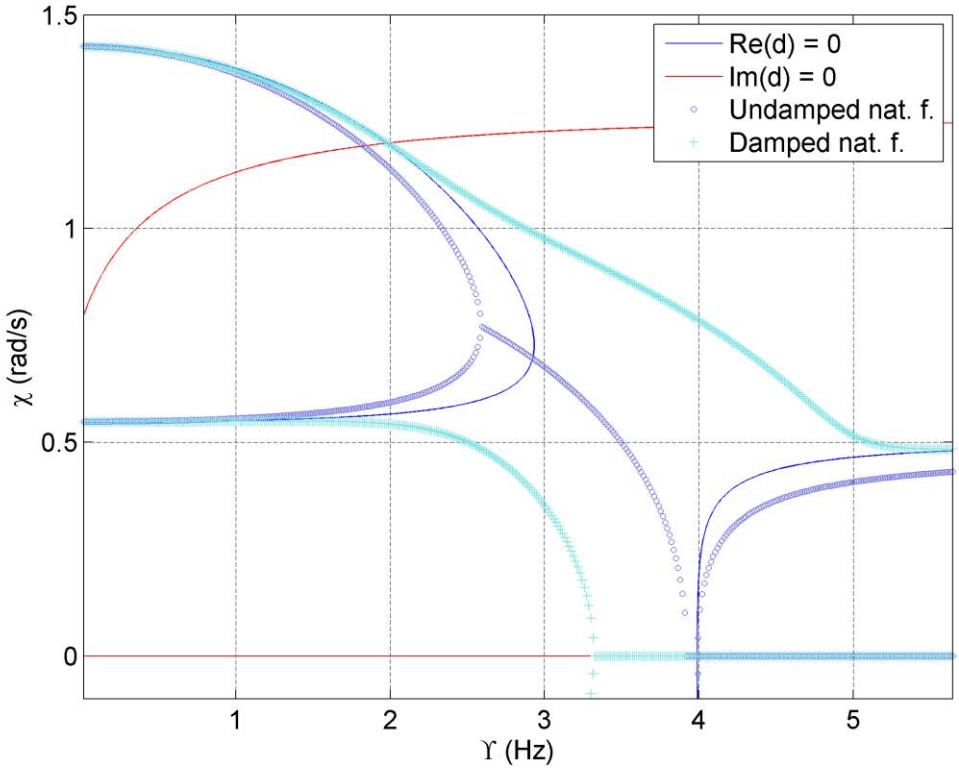


Figure 3.16: Comparison between contour plot and frequency path from modal damping plot, for the section model with quasisteady Theodorsen aerodynamics.

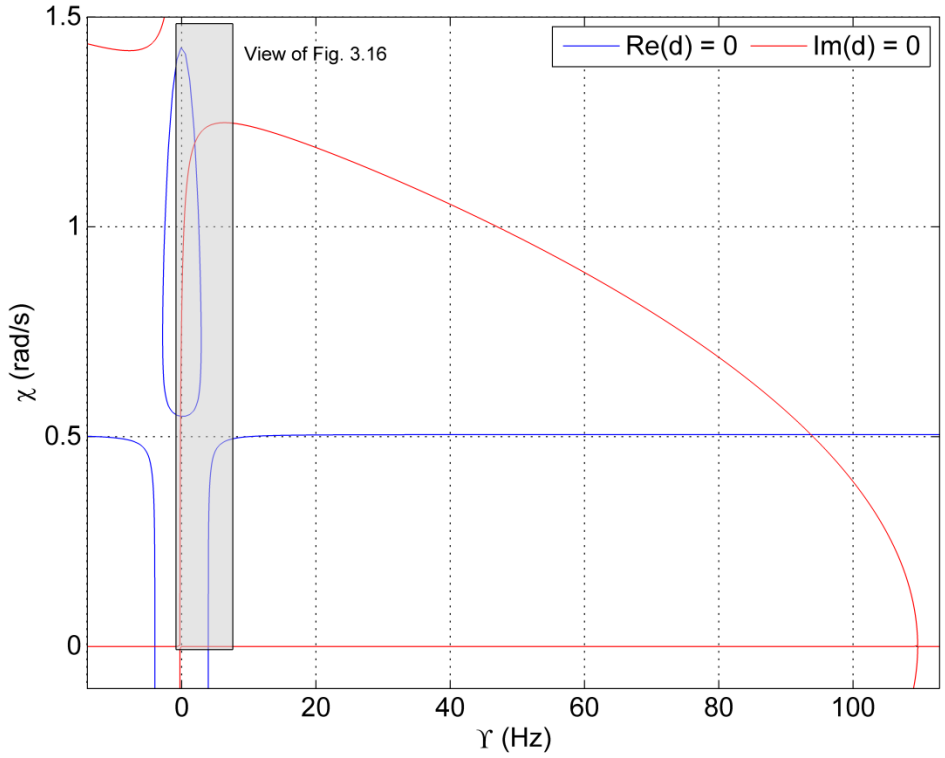


Figure 3.17: Contour plot for Eq. 3.4.4 with wide field of view. The view of Figure 3.26 is indicated

3.4.4 Section model with Theodorsen aerodynamics

The section model with Theodorsen aerodynamics is the most complex model that we will be considering in this thesis, and as such it is worthwhile investigating and visualising its properties before we begin to develop more advanced solvers. Figure 3.18 shows a contour plot, showing flutter points near the origin (physical and nonphysical). Flutter points can be identified at the following locations. Physical flutter point: $\Upsilon = 3.15\ \mathrm{Hz}$, $\chi = 0.890\ \mathrm{rad/s}$. Divergence point: $\Upsilon = 3.99$. Nonphysical flutter points: $(\Upsilon, \chi) = (-0.762, 1.363)$, $(-0.449, 0.567)$, $(-0.180, 0.540)$, $(0.807, -0.578)$, $(-0.279, -0.549)$, $(-3.00, -0.950)$, $(-0.928, -1.40)$. Nonphysical divergence point: $\Upsilon = -3.99\ \mathrm{Hz}$. Figure 3.19 shows a similar contour plot, but with a significantly wider field of view. Two further flutter points can be identified, one physical ($\Upsilon = 93.6\ \mathrm{Hz}$, $\chi = 0.505\ \mathrm{rad/s}$) and the other not ($\Upsilon = 93.5\ \mathrm{Hz}$, $\chi = -0.505$). Note that these two flutter points do indeed have different Υ-ordinates: this is not a feature of the resolution of the contour plot, and can be confirmed by taking a very close-up view of each point. The physical flutter point must represent a restabilisation, as by this point both of the system's two modes have destabilised. For good measure, Figure 3.21 shows the contour plot in τ-λ coordinates. The same flutter points may be identified.

Two features of these contour plots may immediately be noted. The first is the highly oscillatory behaviour of the contour plot near the χ- or λ-axis. A detailed view of these oscillations is provided in Figure 3.20. This behaviour is due to the nature of Theodorsen's function, $C(\kappa)$, at negative κ. Figure 3.22 shows Theodorsen's function. For $\kappa < 0$, $C(\kappa)$ becomes sinusoidal: this is not widely recognised, as $C(\kappa)$ is seldom evaluated at negative κ because such κ is unphysical. Because $\tau = 1/\kappa$ and $\Upsilon = \chi/\kappa$, when $\tau \to 0^-$ or $\Upsilon \to 0^-$ the frequency of oscillation increases boundlessly. Conversely, for the whole of the left half-plane below about $\tau = -0.5$, Theodorsen's function only undergoes one oscillation. This is shown in Figure 3.23. It is the oscillatory behaviour shown in Figure 3.23 that is the direct cause of the oscillations in Figure 3.20 and thus Figure 3.18 and Figure 3.21. It is only by choosing one of the eigenvalue parameters as κ (or a nonnegative power thereof) that this oscillatory behaviour can be prevented. However, forms in κ are less useful as they do not become second-order polynomial when $C(\kappa)$ is held constant (as the Υ-χ and τ-λ forms do).

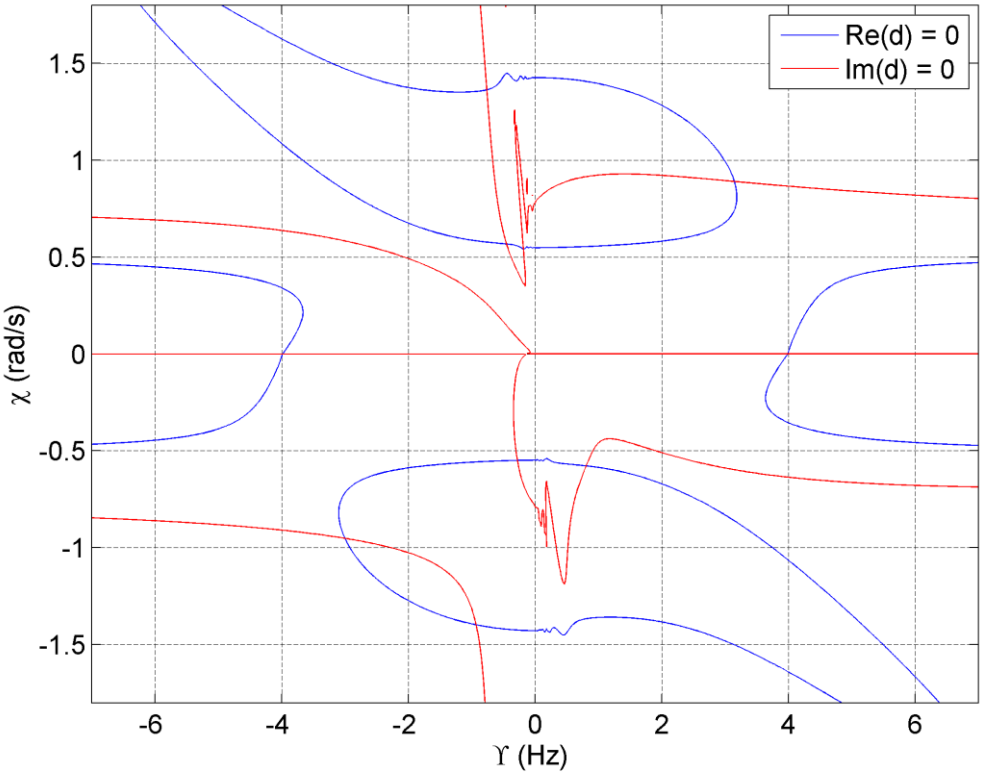


Figure 3.18: Contour plot of section model with Theodorsen aerodynamics (Υ-χ form).

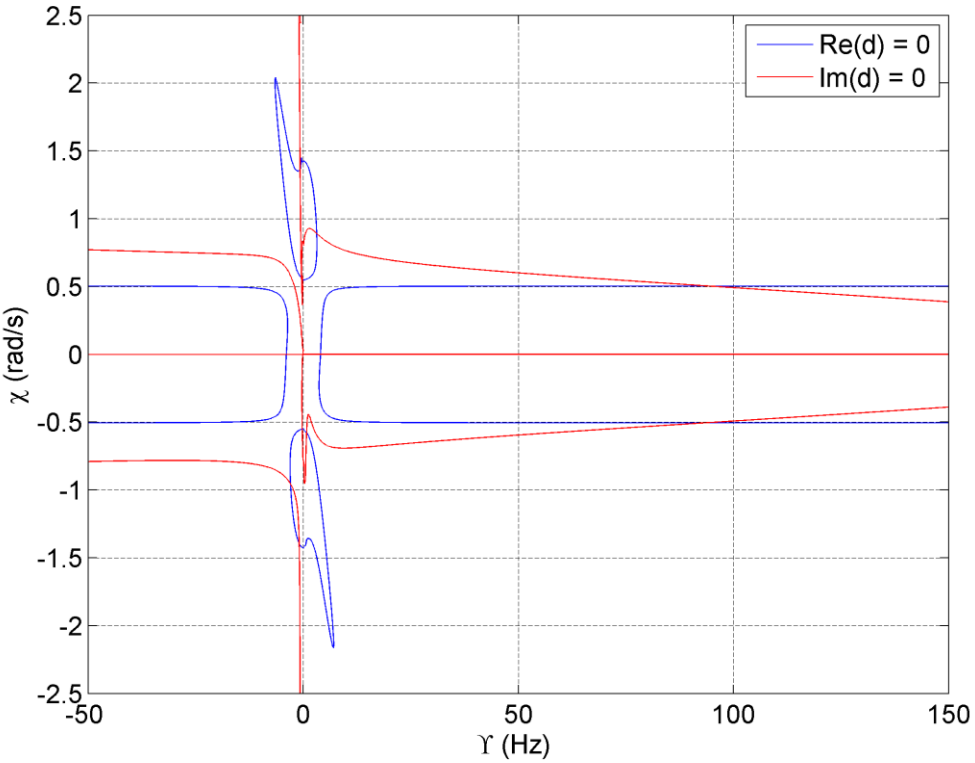


Figure 3.19: Contour plot of section model with Theodorsen aerodynamics (Υ-χ form). A wider view of Figure 3.18.

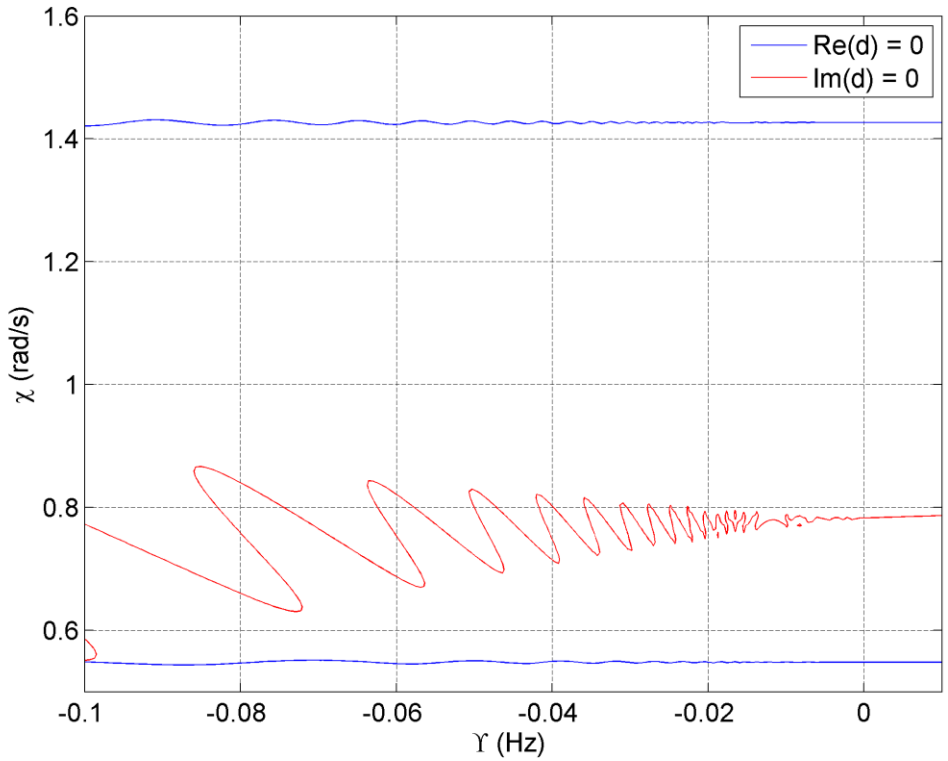


Figure 3.20: Detail of the oscillatory behaviour in Figure 3.18.

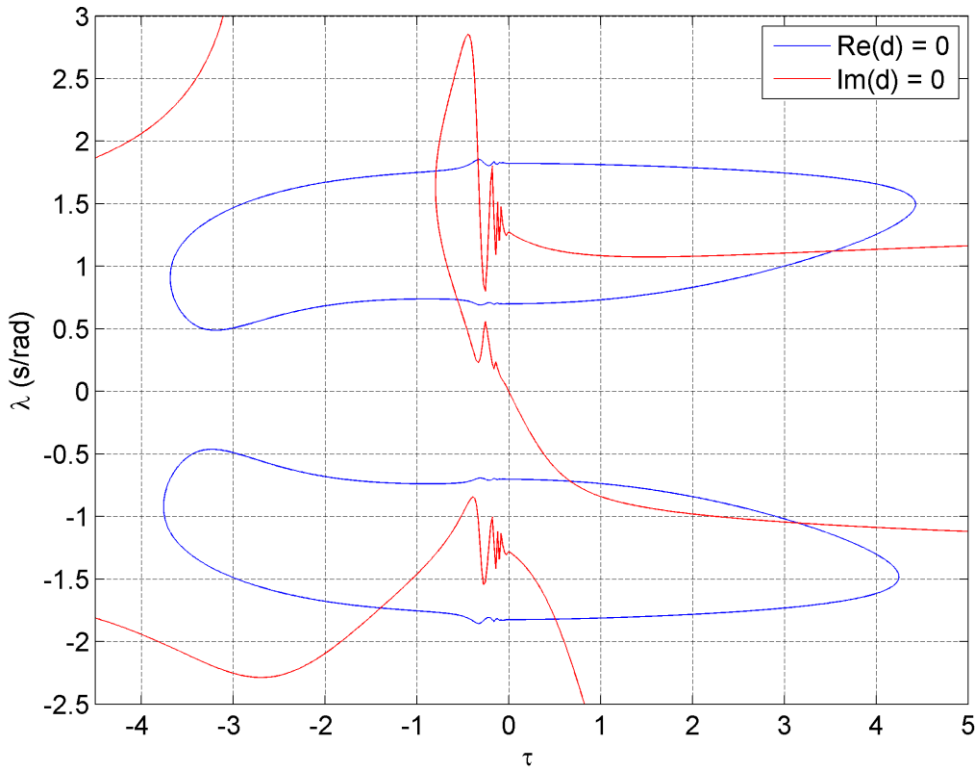


Figure 3.21: Contour plot of section model with Theodorsen aerodynamics (τ-λ form).

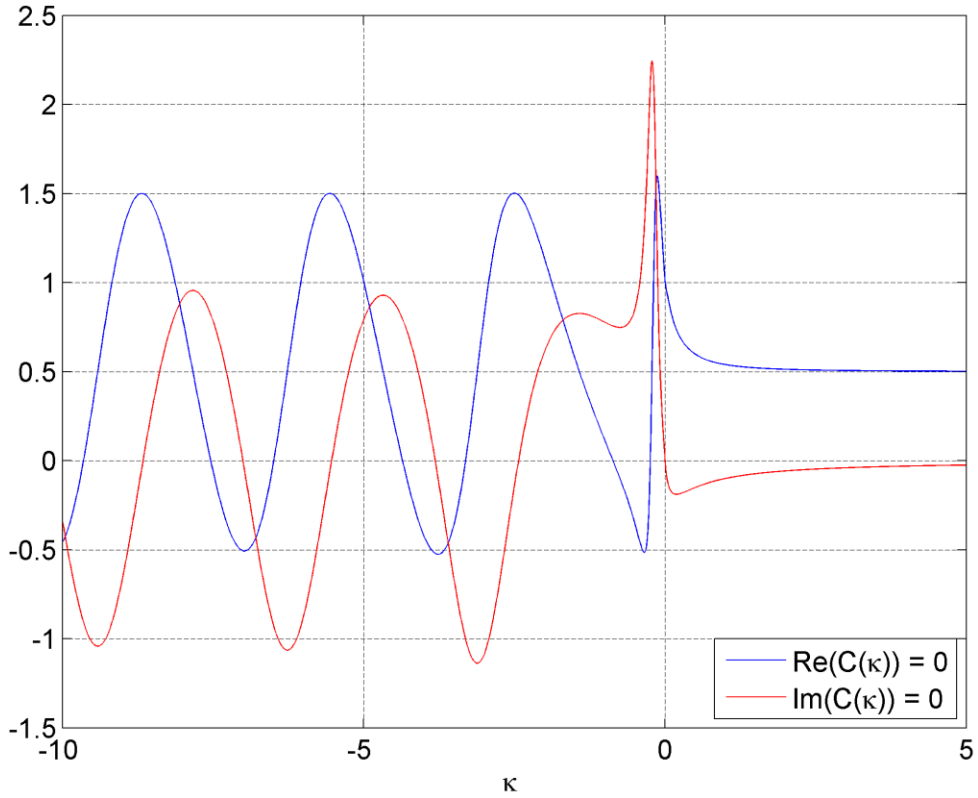


Figure 3.22: Theodorsen's function $C(\kappa)$.

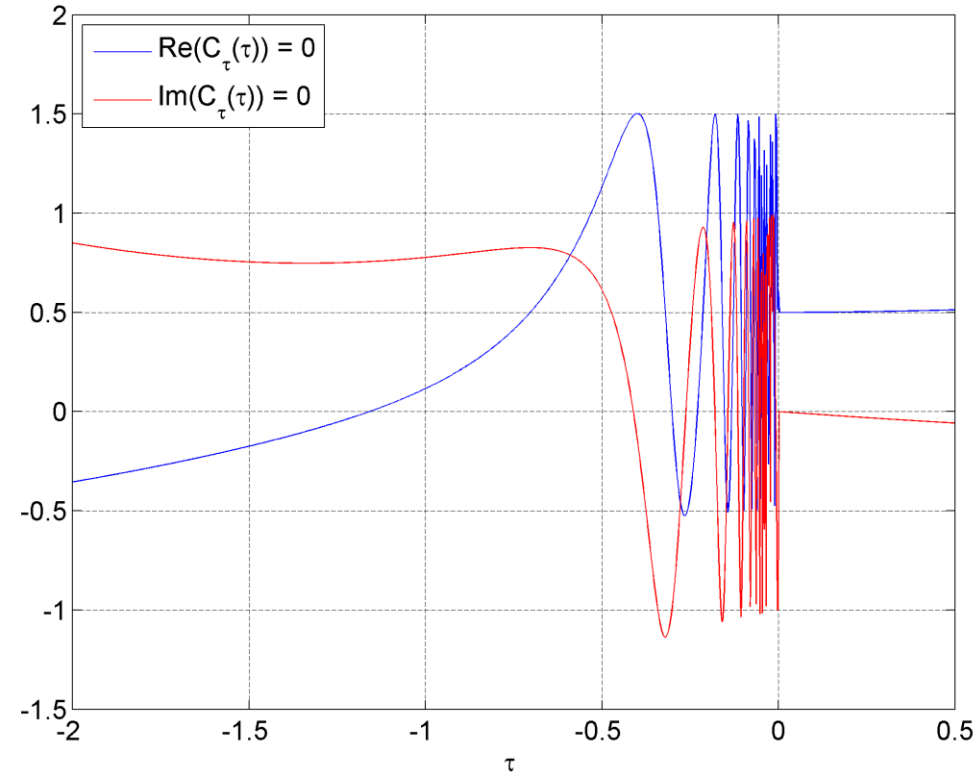


Figure 3.23: Theodorsen's function in τ, $C_\tau(\tau) = C(1/\tau)$.

Another interesting (but more subtle) feature can be also observed in Figure 3.18. At the divergence point at $\Upsilon = \pm 3.99\ \mathrm{Hz}$, the real contour is non-differentiable. This is more obvious on a detailed view of the divergence point (Figure 3.24). Like the oscillations, this is not simply a feature of the real contours, but a property of the matrix function. Figure 3.25 shows the 2-norm of the system matrix,

$$\mathrm{D} = (\mathrm{M}_0 + \mathrm{G}_2)\chi^2 + \left(\mathrm{G}_1 + \mathrm{G}_2 C\left(\frac{\chi}{\Upsilon}\right)\right)\Upsilon\chi + \mathrm{G}_3 C\left(\frac{\chi}{\Upsilon}\right)\Upsilon^2 - \mathrm{D}_0\chi - \mathrm{K}_0 \qquad (3.4.6)$$

as a function of χ, for a set of Υ-values. The non-differentiability at $\chi = 0$ can be plainly seen. This discontinuity is again a feature of Theodorsen's function, and arises from the fact that the Hankel function $H_0^{(2)}(\kappa)$ is discontinuous at $\kappa = 0$. Its effect on the imaginary contours is less marked, but can be observed when the one imaginary contour that passes from the upper half plane to the lower does so, near the origin (see Figure 3.18).

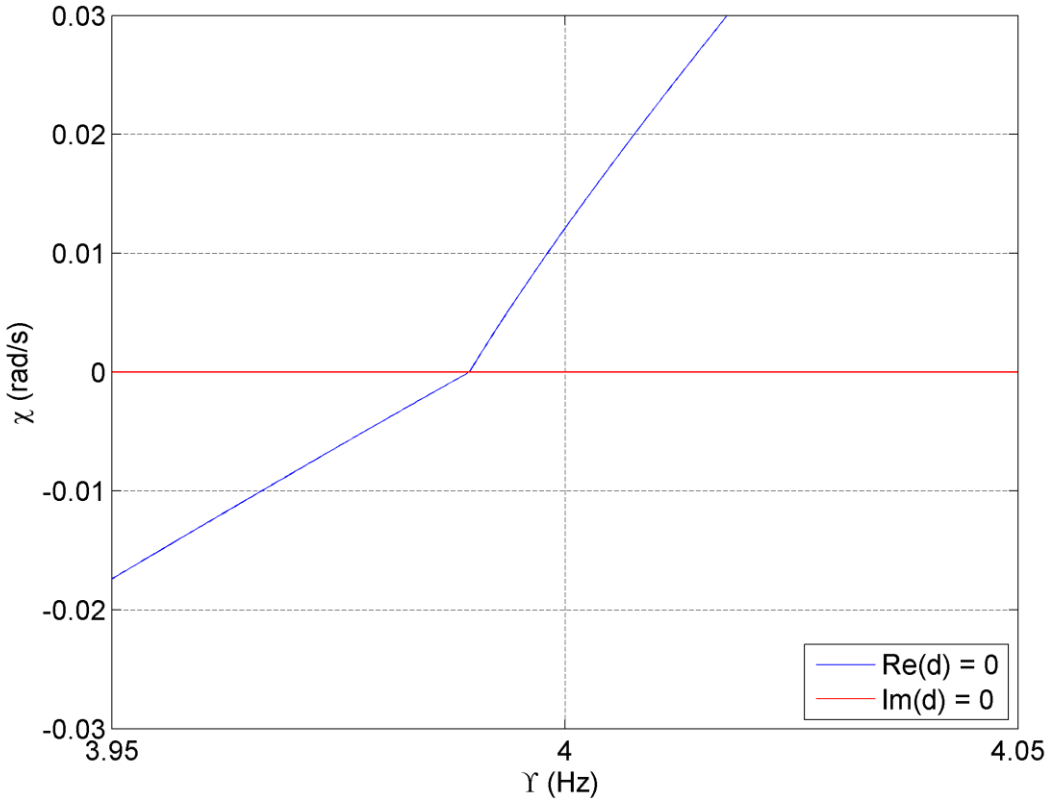


Figure 3.24: Non-differentiable behaviour in the contour plot of the section model with Theodorsen aerodynamics, in the vicinity of the divergence point.

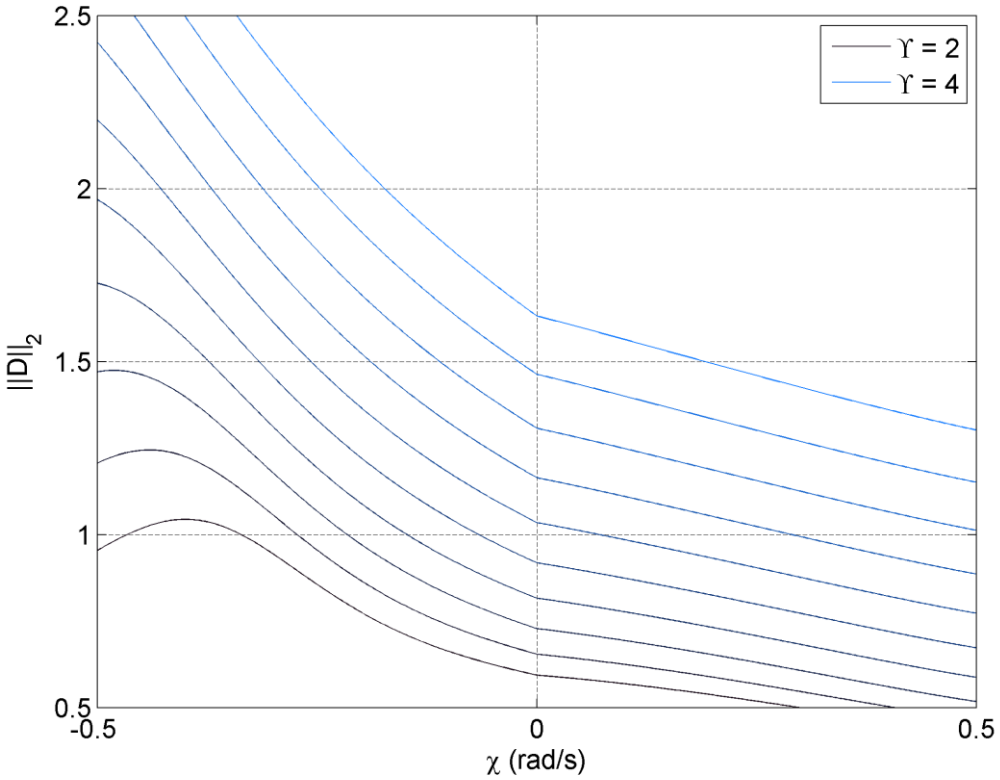


Figure 3.25: Non-differentiable behaviour in the $\|\mathrm{D}\|_2$ in the vicinity of the divergence point.

3.5 CONCLUDING REMARKS

In this chapter we have presented methods for visualising (and thus computing) the flutter points of aeroelastic problems. The methods we have presented are of course very general, and apply generally to instability problems. The modal damping and root locus methods are well-known, and have been applied both in aeroelasticity and in other fields. The reverse root locus and modal damping methods are entirely new, though they are slightly academic and only useful for certain classes of systems. Nevertheless they represent a unique insight that a multiparameter perspective can provide, and it is very interesting that they have never been considered before. It would appear that the interchangeability of the modal frequency and airspeed parameters in these search-based methods (based on the concept that both are equally valid eigenvalues of the multiparameter problem) has not been previously recognised.

But of far greater importance is the contour plot method. This is the most significant novel intellectual contribution of the chapter. Several three-dimensional searches (in real and

imaginary parts of modal frequency, and real airspeed) have been presented in the literature (e.g. [10]), but the concept that a two-dimensional search will suffice for computing the flutter points appears to have ben unrecognised. Moreover, even this two-dimensional section can give useful information as to the three-dimensional behaviour of the system, as we have shown. The greatest advantage of the contour plot method is its applicability to an extremely wide class of systems. It does not require the problem to have any defined matrix structure, and it can handle systems with point discontinuities, poles, or even large areas where the matrix function is not defined. When the system has a continuous spectrum it is necessary to tread carefully in the interpretation, especially because MATLAB's contour function may itself have difficulties. However, the method itself is still sound. For example, Figure 3.8 correctly indicates that every point not in the range $2.60\ \mathrm{Hz} < \Upsilon < 3.99\ \mathrm{Hz}$ is actually a flutter point. The contour plot is essentially the ultimate general unstructured solver. Its only significant disadvantage is the scaling properties of the determinant, which make it increasingly unsuitable for large or poorly-scaled systems. However, there are modifications we could make to mitigate this problem – for example, normalising each row of the determinant, which changes the determinant value but leaves its sign unchanged[2].

Apart from simply devising these methods, we have presented several numerical experiments using them. These experiments use systems we defined in Chapter 2. They function both as assessments of the effectiveness of these visualisation methods, and also as bases on which we will develop other algorithms for the solution of these problems (in the following chapters). The experiments also elucidate the relationship between the methods we have been proposing – most interestingly, the relationship between the modal damping curves and the real contours of the system. In general we found that the contour plot was the most effective method for visualising system behaviour – particularly because of its generality and ease of implementation. It also allowed us to visualise all information necessary for the definition of a flutter point on a single graph.

[2] This follows from the fact that the multiplying any row of a matrix by a scalar multiplies the matrix's determinant by that same scalar, and that any norm must by definition be nonnegative [11].

Lastly, we should note that we have not actually used any specific methods from multiparameter spectral theory in this chapter, but only the general concept that the airspeed parameter is as equal an eigenvalue as the modal frequency parameter. In some ways it might have been better to present this material after the current Chapter 6 (the unstructured solvers) since these visualisation methods do apply to semi-structured or unstructured problems. However, we will need these visualisation methods to validate (and indeed, to visualise) the results that we will develop in later chapters. However, irrespective of the ordering of this material, the visualisation methods that we have presented – particularly the contour plot – will be of interest to other practitioners and researchers in aeroelasticity.

3.6 REFERENCES

[1] Hansen, M. H., 2004, "Aeroelastic stability analysis of wind turbines using an eigenvalue approach," Wind Energy, **7**(2), pp. 133–143.

[2] Hodges, D. H., and Pierce, G. A., 2011, Introduction to Structural Dynamics and Aeroelasticity, Cambridge University Press, New York, New York State, USA.

[3] Bisplinghoff, R. L., Ashley, H., and Halfman, R. L., 1957, Aeroelasticity, Addison-Wesley, Reading, Massachusetts, USA.

[4] Nayfeh, A. H., Ghommem, M., and Hajj, M. R., 2012, "Normal form representation of the aeroelastic response of the Goland wing," Nonlinear Dynamics, **67**(3), pp. 1847–1861.

[5] Patil, M., Hodges, D., and Cesnik, C., 1998, "Nonlinear aeroelastic analysis of aircraft with high-aspect-ratio wings," American Institute of Aeronautics and Astronautics, Long Beach, California, USA.

[6] Sarkar, P. P., Jones, N. P., and Scanlan, R. H., 1994, "Identification of Aeroelastic Parameters of Flexible Bridges," Journal of Engineering Mechanics, **120**(8), pp. 1718–1742.

[7] Xu, F. Y., Chen, X. Z., Cai, C. S., and Chen, A. R., 2012, "Determination of 18 Flutter Derivatives of Bridge Decks by an Improved Stochastic Search Algorithm," Journal of Bridge Engineering, **17**(4), pp. 576–588.

[8] Stærdahl, J. W., Sørensen, N. N., and Nielsen, S. R., 2007, "Aeroelastic stability of suspension bridges using CFD," Proceedings of the 2007 IASS Symposium, Venice, Italy.

[9] Pons, A., and Gutschmidt, S., 2014, “Lower-Wing Flutter in Biplanes,” Proceedings of the 8th European Nonlinear Dynamics Conference (ENOC2014), Vienna, Austria.

[10] Liska, S., and Dowell, E. H., 2009, “Continuum Aeroelastic Model for a Folding-Wing Configuration,” AIAA Journal, **47**(10), pp. 2350–2358.

[11] Anton, H., 1977, Elementary Linear Algebra, John Wiley & Sons, New York, New York State, USA.

Chapter 4
Structured systems

4.1 INTRODUCTION

In Chapter 2 we introduced several structured systems: systems with an explicit analytical dependence on the eigenvalue parameters. We will be primarily concerned with systems which have a polynomial dependence on the eigenvalue parameters. Such systems can be solved by a number of multiparameter solvers. It is these solvers that we will investigate in this chapter: both as a method of solving the flutter problem we have just presented and as a basis for some of the more advanced methods that we will discuss in later chapters.

Note that the much of the purely mathematical material that is presented in this chapter has been developed by previous authors: we do not make many mathematical advances over existing literature. Where these occur they are noted. The significant novel intellectual contribution in this chapter is an application of these methods to the solution of various aeroelastic stability problems. The methods we will investigate in this chapter will also form the basis for most of work later on in this thesis, where we make contributions both to aeronautical engineering and the abstract study of multiparameter eigenvalue problems.

4.2 LINEAR PROBLEMS

4.2.1 Motivation

Consider again our introductory model, presented in Chapter 2, Section 2.2. This model resulted in the two-parameter eigenvalue problem:

$$\begin{aligned}(\mathrm{A}+\mathrm{B}\lambda+\mathrm{C}\mu)\mathbf{x}&=0\\(\overline{\mathrm{A}}+\overline{\mathrm{B}}\lambda+\overline{\mathrm{C}}\mu)\bar{\mathbf{x}}&=0.\end{aligned}\tag{4.2.1}$$

with

$$\mathrm{A}=\begin{bmatrix}-\omega_h^2 & 0\\ 0 & -r^2\omega_\theta^2\end{bmatrix},\qquad \mathrm{B}=\begin{bmatrix}1 & -r_\theta\\ -r_\theta & r^2\end{bmatrix},\qquad \mathrm{C}=\frac{1}{\mu}\begin{bmatrix}0 & -2\\ 0 & 2\left(\frac{1}{2}+a\right)\end{bmatrix}.\tag{4.2.2}$$

This eigenvalue problem is linear: the system is a linear combination of the two eigenvalues. Such systems can be solved directly via the method of operator determinants [1–3], and it

is this method of solution that we will investigate in this section. This will form a basis for the polynomial direct solution methods that we will investigate later in the chapter.

4.2.2 The Kronecker product

Before we can use the operator determinant method, we must define the Kronecker product. The Kronecker product maps two matrices of potentially different size, $\mathrm{A} \in \mathbb{C}^{n\times m}$ and $\mathrm{B} \in \mathbb{C}^{p\times q}$, to a single matrix of larger size $\mathrm{X} \in \mathbb{C}^{(np)\times(mq)}$, such that

$$\mathrm{A} \otimes \mathrm{B} = \mathrm{X} = \begin{bmatrix} a_{11}\mathrm{B} & \cdots & a_{1m}\mathrm{B} \\ \vdots & \ddots & \vdots \\ a_{n1}\mathrm{B} & \cdots & a_{nm}\mathrm{B} \end{bmatrix} \tag{4.2.3}$$

where $A = \left(a_{ij}\right)_{ij}$. The Kronecker product is bilinear and associative (k is a scalar)

$$\begin{gathered} \mathrm{A} \otimes (\mathrm{B} + \mathrm{C}) = \mathrm{A} \otimes \mathrm{B} + \mathrm{A} \otimes \mathrm{C} \\ (\mathrm{A} + \mathrm{B}) \otimes \mathrm{C} = \mathrm{A} \otimes \mathrm{C} + \mathrm{B} \otimes \mathrm{C} \\ \mathrm{A} \otimes (\mathrm{B} \otimes \mathrm{C}) = (\mathrm{A} \otimes \mathrm{B}) \otimes \mathrm{C} \\ (k\mathrm{A}) \otimes \mathrm{B} = k(\mathrm{A} \otimes \mathrm{B}) = \mathrm{A} \otimes (k\mathrm{B}) \end{gathered} \tag{4.2.4}$$

but not necessarily commutative

$$\mathrm{A} \otimes \mathrm{B} \neq \mathrm{B} \otimes \mathrm{A} \tag{4.2.5}$$

It also satisfies another relation known as the Mixed Product Property

$$(\mathrm{A} \otimes \mathrm{B}) \times (\mathrm{C} \otimes \mathrm{D}) = (\mathrm{A} \times \mathrm{C}) \otimes (\mathrm{B} \times \mathrm{D}) \tag{4.2.6}$$

where $\times$ is the standard matrix multiplication operation. Of course, Eq. 4.2.6 only holds if the sizes of the matrices are such that the various matrix multiplications can be carried out: a necessary condition for this is $\mathrm{A} \in \mathbb{C}^{n\times m}$, $\mathrm{B} \in \mathbb{C}^{p\times q}$, $\mathrm{C} \in \mathbb{C}^{m\times k}$, $\mathrm{D} \in \mathbb{C}^{q\times r}$. Further details on the properties of the Kronecker product may be found in [4,5].

4.2.3 The operator determinant method for two-parameter systems

Consider a slightly more general version of Eq. 4.2.1:

$$\begin{aligned} (\mathrm{A}_1 + \mathrm{B}_1\lambda + \mathrm{C}_1\mu)\mathbf{x} &= 0, \\ (\mathrm{A}_2 + \mathrm{B}_2\lambda + \mathrm{C}_2\mu)\mathbf{y} &= 0. \end{aligned} \tag{4.2.7}$$

While we could specify $\mathrm{A}_2 = \overline{\mathrm{A}_1}$, we will later be interested in linear systems with more than two parameters and so we will keep to the more general form for the purposes of

introduction. Note also that the two equations need not be the same size. Post-multiplying the first equation in Eq. 4.2.7 system by $C_2\mathbf{y}$ and premultiplying the second by $C_1\mathbf{x}$, we have

$$\begin{aligned}(\mathrm{A}_1 + \mathrm{B}_1\lambda + \mathrm{C}_1\mu)\mathbf{x} \otimes (\mathrm{C}_2\mathbf{y}) = 0,\\ (\mathrm{C}_1\mathbf{x}) \otimes (\mathrm{A}_2 + \mathrm{B}_2\lambda + \mathrm{C}_2\mu)\mathbf{y} = 0.\end{aligned} \tag{4.2.8}$$

These two equations are both equal to zero so we may equate them. After cancelling the terms in μ, the expression can be manipulated into:

$$(\mathrm{C}_1 \otimes \mathrm{A}_2 - \mathrm{A}_1 \otimes \mathrm{C}_2)(\mathbf{x} \otimes \mathbf{y}) + \lambda(\mathrm{B}_1 \otimes \mathrm{C}_2 - \mathrm{C}_1 \otimes \mathrm{B}_2)(\mathbf{x} \otimes \mathbf{y}) = 0 \tag{4.2.9}$$

Defining the operator determinants

$$\begin{aligned}\Delta_0 &= \mathrm{B}_1 \otimes \mathrm{C}_2 - \mathrm{C}_1 \otimes \mathrm{B}_2\\ \Delta_1 &= \mathrm{C}_1 \otimes \mathrm{A}_2 - \mathrm{A}_1 \otimes \mathrm{C}_2\\ \Delta_2 &= \mathrm{A}_1 \otimes \mathrm{B}_2 - \mathrm{B}_1 \otimes \mathrm{A}_2\end{aligned} \tag{4.2.10}$$

and an enlarged eigenvector

$$\mathbf{z} = \mathbf{x} \otimes \mathbf{y} \tag{4.2.11}$$

this becomes

$$\Delta_1\mathbf{z} = \lambda\Delta_0\mathbf{z} \tag{4.2.12}$$

which is a generalised eigenvalue problem, in the single parameter λ. Solvers for generalised eigenvalue problem are very well known; MATLAB has an inbuilt QZ algorithm in the command `eig(A, B)`. By multiplying the first and second equations of Eq. 4.2.7 by $\mathrm{B}_2\mathbf{y}$ and $\mathrm{B}_1\mathbf{x}$ respectively, we can also show that:

$$\Delta_2\mathbf{z} = \mu\Delta_0\mathbf{z}. \tag{4.2.13}$$

However, it is only necessary to solve one of Eq. 4.2.12 or Eq. 4.2.13: once one has been solved (say, Eq. 4.2.13, yielding the μ-coordinates for all the eigenvalue points), then its solutions can be substituted back into either equation of Eq. 4.2.1, which yields another generalised eigenvalue problem. This is both computationally cheaper (the system is size $n \times n$ and not $n^2 \times n^2$), and it also allows us to determine the original modeshape ($\mathbf{x}$ or $\mathbf{y}$), rather than $\mathbf{z} = \mathbf{x} \otimes \mathbf{y}$.

In the case of Eq. 4.2.1, the operator determinants become

$$\begin{aligned}\Delta_0 &= \mathrm{B} \otimes \overline{\mathrm{C}} - \mathrm{C} \otimes \overline{\mathrm{B}} \\ \Delta_1 &= \mathrm{C} \otimes \overline{\mathrm{A}} - \mathrm{A} \otimes \overline{\mathrm{C}} \\ \Delta_2 &= \mathrm{A} \otimes \overline{\mathrm{B}} - \mathrm{B} \otimes \overline{\mathrm{A}}.\end{aligned} \tag{4.2.14}$$

If the original system is of size $\mathrm{A} \in \mathbb{C}^{n \times n}$ then $\Delta_i = \mathbb{C}^{n^2 \times n^2}$. We can thus devise the following direct solver algorithm.

Algorithm 4.1 – simple direct solver for the flutter points of $(\mathrm{A} + \mathrm{B}\lambda + \mathrm{C}\mu)\mathbf{x} = 0$

1	initialise A, B and C
2	compute $\Delta_0 = \mathrm{B} \otimes \overline{\mathrm{C}} - \mathrm{C} \otimes \overline{\mathrm{B}}$ $\Delta_2 = \mathrm{A} \otimes \overline{\mathrm{B}} - \mathrm{B} \otimes \overline{\mathrm{A}}$
3	compute the set of real eigenvalues $\{\mu_k\} \in \mathbb{R}$ of $\Delta_2 \mathbf{z} = \mu \Delta_0 \mathbf{z}$
4	**for** each $\mu_k \in \{\mu_k\}$
5	compute the set of real eigenvalues $\{\lambda_i\}_{(k)} \in \mathbb{R}$ and eigenvectors $\{\mathbf{x}_i\}_{(k)} \in \mathbb{C}^n$ of $(\mathrm{A} + \mathrm{B}\lambda + \mathrm{C}\mu_k)\mathbf{x} = 0$
6	**end for**
7	return $\{\mu_k\}, \{\{\lambda_i\}_{(k)}\}, \{\{\mathbf{x}_i\}_{(k)}\}$

One important caveat of the operator determinant approach is that the matrix Δ_0 must not be singular: the robust proof that the eigenvalues of Eq. 4.2.12 and Eq. 4.2.13 coincide with those of Eq. 4.2.1 only holds if this is the case [2,6,7]. A linear multiparameter eigenvalue problem with singular Δ_0 is said to be singular multiparameter eigenvalue problem. Unfortunately, we will find that a large proportion of the linear flutter problems that arise in the study of aircraft aeroelasticity are in fact singular. This includes Eq. 4.2.1:

$$\begin{aligned}\Delta_0 &= \mathrm{B} \otimes \overline{\mathrm{C}} - \mathrm{C} \otimes \overline{\mathrm{B}} \\ &= \begin{bmatrix} 1 & -r_\theta \\ -r_\theta & r^2 \end{bmatrix} \otimes \frac{1}{\mu}\begin{bmatrix} 0 & -2 \\ 0 & 2\left(\frac{1}{2}+a\right) \end{bmatrix} - \frac{1}{\mu}\begin{bmatrix} 0 & -2 \\ 0 & 2\left(\frac{1}{2}+a\right) \end{bmatrix} \otimes \begin{bmatrix} 1 & -r_\theta \\ -r_\theta & r^2 \end{bmatrix} \\ &= \frac{1}{\mu}\begin{bmatrix} 0 & -2 & -2 & 0 \\ 0 & 2\left(\frac{1}{2}+a\right) & 2r_\theta & -2r_\theta\left(\frac{1}{2}+a\right) - 2r^2 \\ 0 & -2r_\theta & 2\left(\frac{1}{2}+a\right) & 2r_\theta\left(\frac{1}{2}+a\right) + 2r^2 \\ 0 & -2r_\theta\left(\frac{1}{2}+a\right) & 2r_\theta\left(\frac{1}{2}+a\right) & 0 \end{bmatrix}\end{aligned} \tag{4.2.15}$$

which is singular by inspection. A solver for singular systems is thus clearly necessary, but until very recently there have been no direct methods for such systems.

4.2.4 The compression algorithm

Recent work by Muhič and Plestenjak [3] proved that the eigenvalues of Eq. 4.2.7 are equivalent to the finite regular eigenvalues of Eq. 4.2.17 and 4.2.18 when Δ_0 is singular. The finite regular eigenvalues of Eq. 4.2.7 are the pairs (λ, μ) such that [2]:

$$\operatorname{rank}(\mathrm{A}_i + \mathrm{B}_i\lambda + \mathrm{C}_i\mu) < \max_{(s,t)\in\mathbb{C}^2} \operatorname{rank}(\mathrm{A}_i + \mathrm{B}_i s + \mathrm{C}_i t) \tag{4.2.16}$$

for $i = 1, 2$. On the basis of this proof, they devised a set of algorithms which would extract the common regular part of the singular matrix pencils[1] $\Delta_{1\mathrm{s}} - \lambda\Delta_{0\mathrm{s}}$ and $\Delta_{2\mathrm{s}} - \lambda\Delta_{0\mathrm{s}}$. This common regular part is represented by two smaller non-singular matrix pencils ($\Delta_1 - \lambda\Delta_0$ and $\Delta_2 - \lambda\Delta_0$), which can then be solved using the operator determinant method as presented in Section 4.2.2. In practical terms, this can be seen as a compression of the singular operator determinants into smaller full-rank matrices. These algorithms are presented in [3], and we will not detail them here as the theory involved is beyond the scope of this thesis. Algorithm 4.2 presents the direct solver of Algorithm 4.1 with a compression stage implemented. We are grateful to Muhič and Plestenjak [3] for providing code for the compression algorithms. During the writing of this thesis it was noted that Muhič and Plestenjak have published their multiparameter eigenvalue code (including this compression algorithm) on the MATLAB file exchange under the name 'MultiParEig' [8]. Most of the solvers in this thesis were developed without recourse to this code.

Algorithm 4.2 – direct solver with compression

1	initialise A, B and C
2	compute $\Delta_{0\mathrm{s}} = \mathrm{B} \otimes \overline{\mathrm{C}} - \mathrm{C} \otimes \overline{\mathrm{B}}$ $\Delta_{1\mathrm{s}} = \mathrm{C} \otimes \overline{\mathrm{A}} - \mathrm{A} \otimes \overline{\mathrm{C}}$ $\Delta_{2\mathrm{s}} = \mathrm{A} \otimes \overline{\mathrm{B}} - \mathrm{B} \otimes \overline{\mathrm{A}}$

[1] A matrix pencil: a matrix-valued function $\mathrm{F}_n(\lambda) = \mathrm{A}_0 + \mathrm{A}_1\lambda + \mathrm{A}_2\lambda^2 + \cdots + A_n\lambda^n$ with order n and coefficient matrices A_i. The problem of finding the eigenvalues of a given polynomial problem is equivalent to the problem of finding the roots of determinant of the corresponding matrix pencil. It is from the latter perspective that Muhič and Plestenjak [3] approach their singular solver.

3	compress Δ_{0s}, Δ_{1s}, and Δ_{2s} into Δ_0, Δ_1, and Δ_2
4	compute the set of real eigenvalues $\{\mu_k\} \in \mathbb{R}$ of $$\Delta_2 \mathbf{z} = \mu \Delta_0 \mathbf{z}$$
5	**for** each $\mu_k \in \{\mu_k\}$
6	compute the set of real eigenvalues $\{\lambda_i\}_{(k)} \in \mathbb{R}$ and eigenvectors $\{\mathbf{x}_i\}_{(k)} \in \mathbb{C}^n$ of $$(\mathrm{A} + \mathrm{B}\lambda + \mathrm{C}\mu_k)\mathbf{x} = 0$$
7	**end for**
8	return $\{\mu_k\}$, $\left\{\{\lambda_i\}_{(k)}\right\}$, $\left\{\{\mathbf{x}_i\}_{(k)}\right\}$

4.2.5 Computational complexity

The computational complexity (or time complexity / computational efficiency) of a given algorithm is defined as the number of floating point operations (flops) required to execute the algorithm [9,10]. In the absence of computer overheads, etc., the computational complexity should be directly proportional to the wall-clock time required to execute the algorithm. The two-parameter operator determinant method, without compression, is known to have computational complexity of $\mathcal{O}(n^6)$, under the assumption that both equations are of the same size (n) [6,11,12]. This large complexity arises from solving the generalised eigenvalue problem (an $\mathcal{O}(n^3)$ process by the QZ algorithm [13]) with operator determinants of size $n^2 \times n^2$.

4.2.6 Numerical experiments

When we attempt to apply our new-found operator determinant method to our simple introductory model, Eq. 4.2.1 with parameter values from Chapter 2, we come to a problem. The problem is that the system is entirely undamped, and so every airspeed value up to the first physical flutter point ($\Upsilon_F = 2.600$ Hz) is a point of neutral stability. The system thus has a continuous spectrum in Υ-χ space, and the operator determinant method cannot identify the actual physical flutter points. In Algorithm 3.2, this manifests itself as compressed operator determinants (Δ_0, Δ_1, and Δ_2) that are empty arrays. We saw the same phenomenon in Chapter 2, where the contour plot is free from any imaginary contours.

There is, however, a numerical trick we can apply to get an answer out of the operator determinant method: we perturb the system with a small complex component. For example, we might modify B to be

$$\mathrm{B} = \begin{bmatrix} 1 + 0.001\iota & -r_\theta \\ -r_\theta & r^2 + 0.001\iota \end{bmatrix} \tag{4.2.17}$$

The small complex component ensures that the system is always damped (however slightly), and so the spectrum is now discrete and consists only of actual transitions from instability to stability or vice versa. Figure 4.1 shows a modal damping plot of the perturbed system, with the flutter points computed by the operator determinant method. Figure 4.2 shows a view of the modal damping paths which has been zoomed in the y-axis. The computed flutter and divergence points match reasonably well to those of the unperturbed system. The flutter point is computed to lie at $\Upsilon_F = 2.52\ \mathrm{Hz}$ and $\chi_F = 0.902\ \mathrm{rad/s}$; the value from the unperturbed modal damping plot is $\Upsilon_F = 2.600\ \mathrm{Hz}$ and $\chi_F = 0.7853\ \mathrm{rad/s}$. The divergence point is computed to lie at $\Upsilon_D = 3.99\ \mathrm{Hz}$, a good match with the exact figure at $3.990\ \mathrm{Hz}$. However, one spurious divergence point is created near $\Upsilon = 4.5$, where one of the branches crosses back into stability.

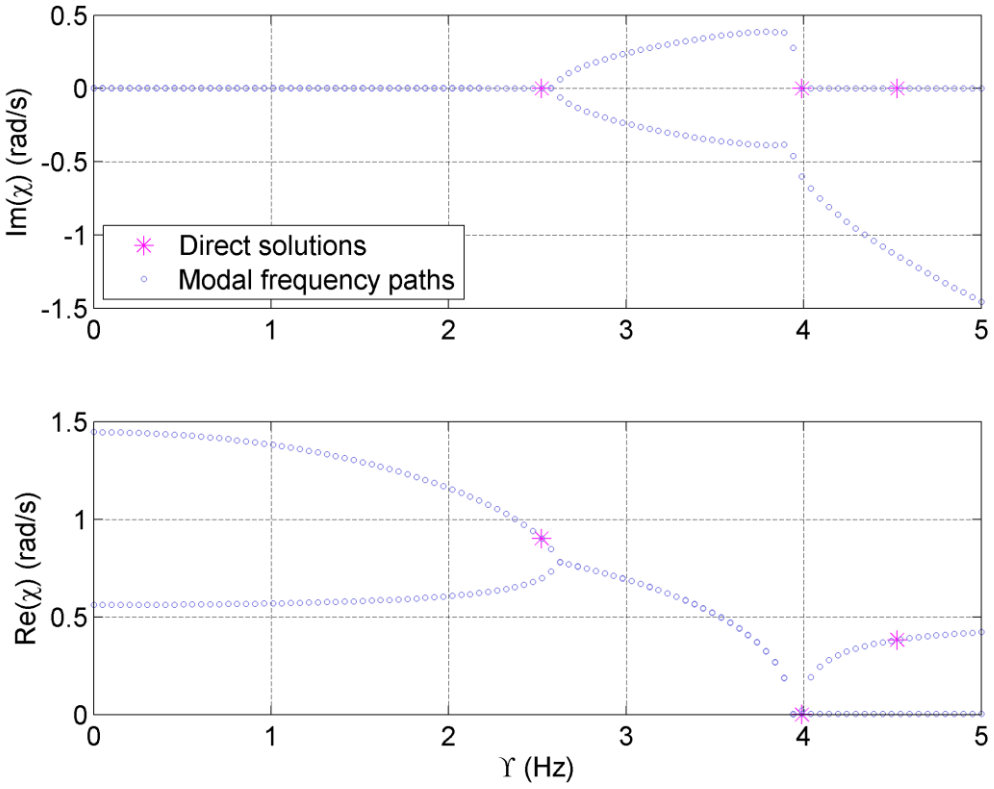


Figure 4.1: Modal damping plot for perturbed introductory system, showing the flutter points computed by the operator determinant method.

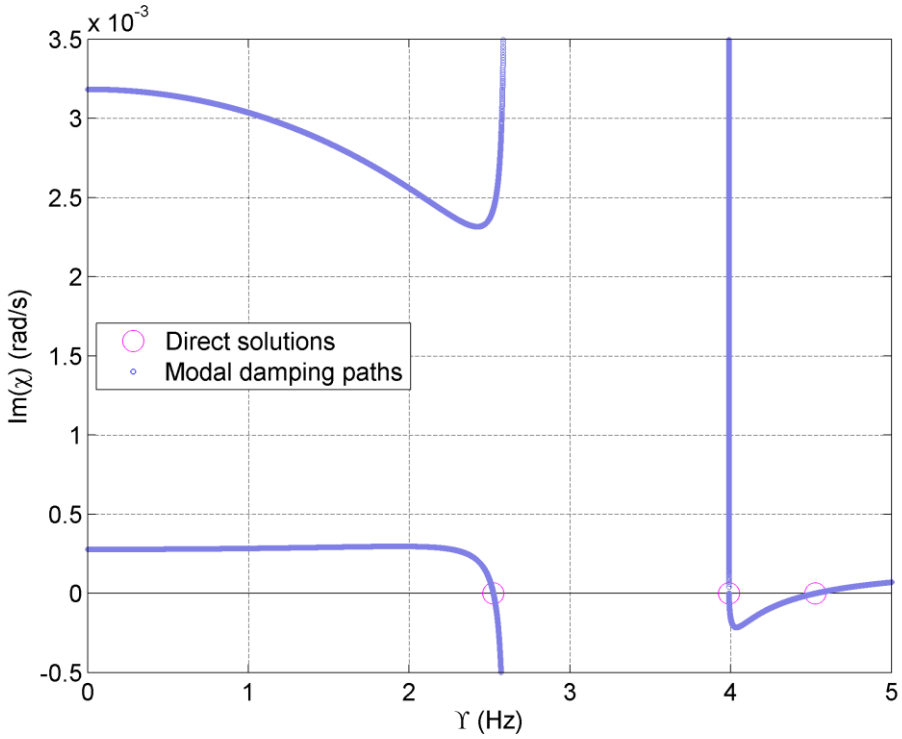


Figure 4.2: Zoomed view of modal damping paths (top plot of Figure 4.1)

While the agreement between the perturbed and exact results look promising, we should note that perturbing the system in this way is numerically irresponsible and is not generally advised. There are a number of complex perturbations which will significantly distort the location of the flutter points. For example, if we instead perturb the aerodynamic forcing matrix as follows

$$\mathrm{C} = \frac{1}{\mu}\begin{bmatrix} 0 & 2 + 0.001\iota \\ 0 & 2\left(\frac{1}{2} + a\right) + 0.001\iota \end{bmatrix} \tag{4.2.18}$$

then the operator determinant method does not locate any useful flutter points at all. Figure 4.3 shows a modal damping plot for this bas perturbation, and Figure 4.4 a view of the modal damping paths which has been zoomed in the y-axis. The flutter point at about $\Upsilon = 1.8\ \mathrm{Hz}$ arises because one of the nearly-undamped branches crosses the imaginary axis at this point: the split between the eigenvalue branches which occurs at $\Upsilon = 2.6\ \mathrm{Hz}$ in the exact system (marking the flutter point) now occurs at $\Upsilon = 0\ \mathrm{Hz}$. Both branches are then slightly negatively damped, until one slowly climbs above $\mathrm{Im}(\chi) = 0$ and becomes the upper branch in the visible branch split at $\Upsilon = 2.6\ \mathrm{Hz}$. The actual branch movement before this point is too small to be made out in Figure 4.3, but is shown in Figure 4.4.

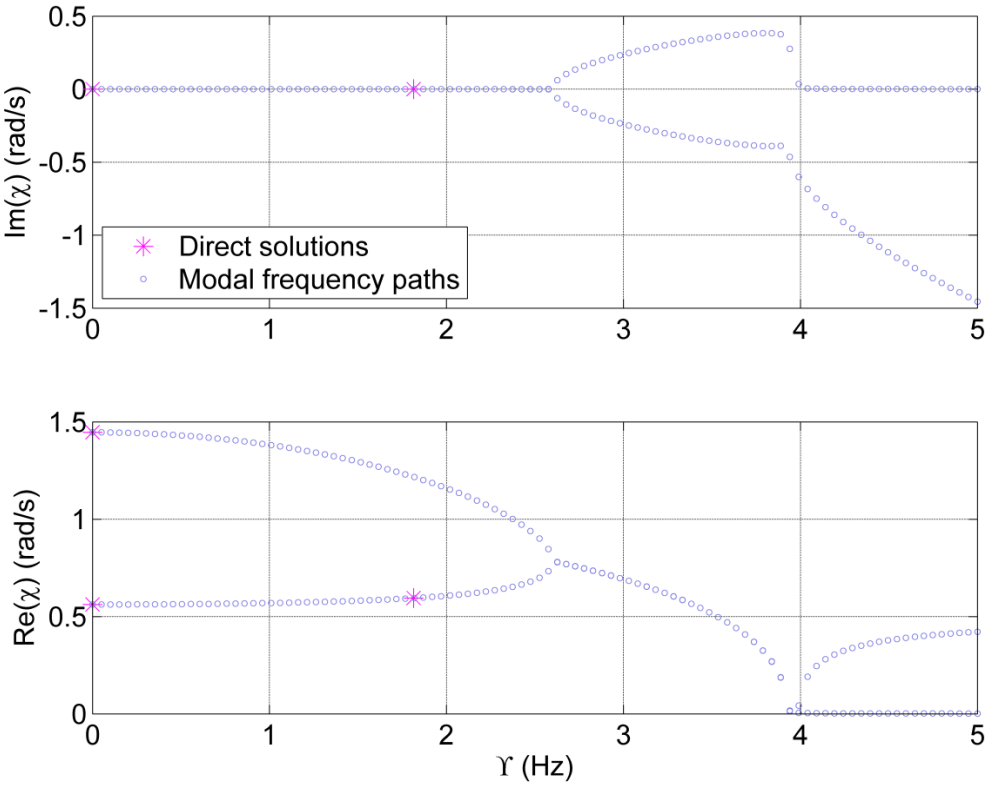


Figure 4.3: Modal damping plot for a badly-perturbed introductory system.

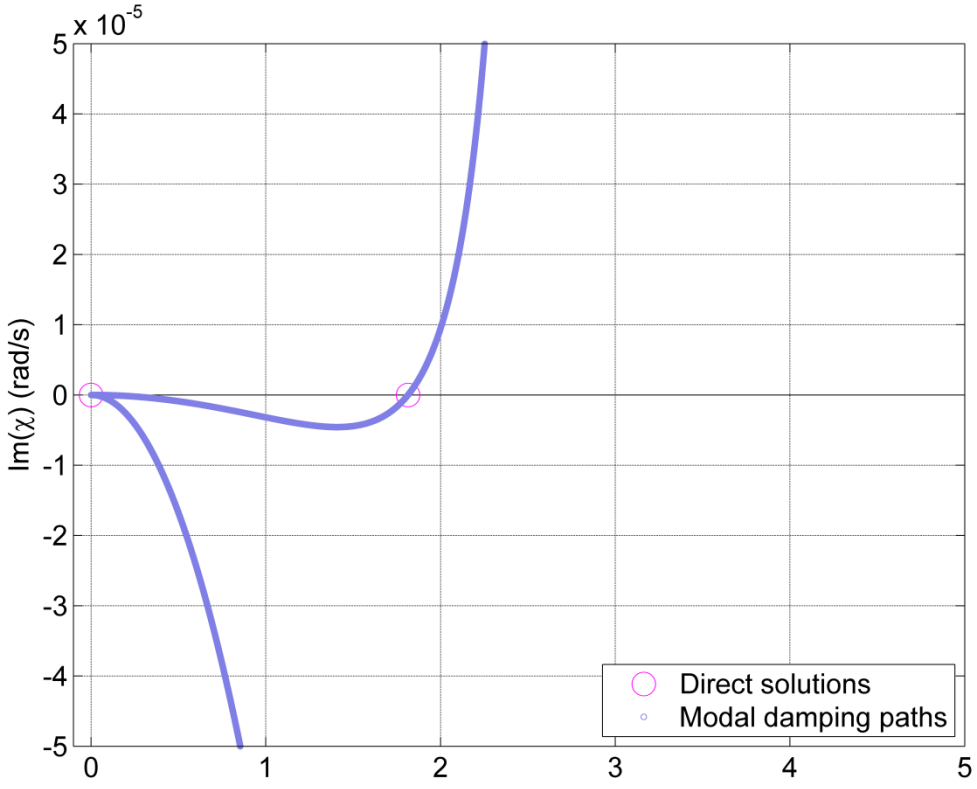


Figure 4.4: Zoomed view of modal damping paths (top plot of Figure 4.2)

The study of the effect of these perturbations on the system's properties, and their use in the operator determinant method, would be an interesting area for further research. However, the study of perturbations will not concern us during the rest of this thesis, as all further problems we consider will already have significant damping; aerodynamic, structural or both.

4.3 QUADRATIC PROBLEMS

4.3.1 Motivation

In this section we will be looking at multiparameter systems which contain quadratic terms in one or more eigenvalue parameters. We introduced three such systems in Chapter 2, all based on the section model with unsteady aerodynamics. The three systems are:

$$\begin{aligned}\big((\mathrm{M}_0+\mathrm{G}_0)+(\mathrm{G}_1+\mathrm{G}_2)\tau+\mathrm{G}_3\tau^2-\mathrm{K}_0\Lambda\big)\mathbf{x}&=\mathbf{0},\\ \big((\overline{\mathrm{M}}_0+\overline{\mathrm{G}}_0)+(\overline{\mathrm{G}}_1+\overline{\mathrm{G}}_2)\tau+\overline{\mathrm{G}}_3\tau^2-\overline{\mathrm{K}}_0\Lambda\big)\bar{\mathbf{x}}&=\mathbf{0}\end{aligned} \tag{4.3.1}$$

which models a section model with quasisteady aerodynamics and no structural damping, and:

$$\begin{aligned}\big((\mathrm{M}_0+\mathrm{G}_0)+(\mathrm{G}_1+\mathrm{G}_2)\tau+\mathrm{G}_3\tau^2-\mathrm{D}_0\lambda-\mathrm{K}_0\lambda^2\big)\mathbf{x}&=\mathbf{0},\\ \big((\overline{\mathrm{M}}_0+\overline{\mathrm{G}}_0)+(\overline{\mathrm{G}}_1+\overline{\mathrm{G}}_2)\tau+\overline{\mathrm{G}}_3\tau^2-\overline{\mathrm{D}}_0\lambda-\overline{\mathrm{K}}_0\lambda^2\big)\bar{\mathbf{x}}&=\mathbf{0}\end{aligned} \tag{4.3.2}$$

$$\begin{aligned}\big((\mathrm{M}_0+\mathrm{G}_0)\chi^2+(\mathrm{G}_1+\mathrm{G}_2)\Upsilon\chi+\mathrm{G}_3\Upsilon^2-\mathrm{D}_0\chi-\mathrm{K}_0\big)\mathbf{x}&=\mathbf{0},\\ \big((\overline{\mathrm{M}}_0+\overline{\mathrm{G}}_0)\chi^2+(\overline{\mathrm{G}}_1+\overline{\mathrm{G}}_2)\Upsilon\chi+\overline{\mathrm{G}}_3\Upsilon^2-\overline{\mathrm{D}}_0\chi-\overline{\mathrm{K}}_0\big)\bar{\mathbf{x}}&=\mathbf{0}\end{aligned} \tag{4.3.3}$$

both of which model a section model with quasisteady aerodynamics and structural damping. The coefficient matrices are defined as

$$\begin{aligned}\mathrm{G}_0&=\frac{1}{\mu}\begin{bmatrix}1 & a\\ a & \left(\frac{1}{8}+a^2\right)\end{bmatrix}, \qquad \mathrm{G}_1=\frac{1}{\mu}\begin{bmatrix}0 & \iota\\ 0 & -\iota\left(\frac{1}{2}-a\right)\end{bmatrix},\\ \mathrm{G}_2&=\frac{1}{\mu}\begin{bmatrix}-2\iota & 2\iota\left(\frac{1}{2}-a\right)\\ -2\iota\left(\frac{1}{2}+a\right) & 2\iota\left(\frac{1}{4}-a^2\right)\end{bmatrix}, \qquad \mathrm{G}_3=\frac{1}{\mu}\begin{bmatrix}0 & 2\\ 0 & 2\left(\frac{1}{2}+a\right)\end{bmatrix},\\ \mathrm{M}_0&=\begin{bmatrix}1 & -r_\theta\\ -r_\theta & r^2\end{bmatrix}, \qquad \mathrm{D}_0=\begin{bmatrix}2\iota\zeta_h\omega_h & 0\\ 0 & 2\iota r^2\zeta_\theta\omega_\theta\end{bmatrix}, \qquad \mathrm{K}_0=\begin{bmatrix}\omega_h^2 & 0\\ 0 & r^2\omega_\theta^2\end{bmatrix}\end{aligned} \tag{4.3.4}$$

These problems can be transformed into linear ones via two different linearisation methods.

4.3.2 Linearisation

Any polynomial multiparameter eigenvalue problem can be made into a linear one via the process of linearisation. Consider first Eq. 4.3.1. To linearise this system, we define two new eigenvectors:

$$\mathbf{q}_1 = \begin{bmatrix} \mathbf{x} \\ \tau\mathbf{x} \end{bmatrix}, \qquad \mathbf{q}_2 = \begin{bmatrix} \bar{\mathbf{x}} \\ \tau\bar{\mathbf{x}} \end{bmatrix}, \tag{4.3.5}$$

or, more compactly, $\mathbf{q}_1 = [\mathbf{x}; \tau\mathbf{x}]$, $\mathbf{q}_2 = [\bar{\mathbf{x}}; \tau\bar{\mathbf{x}}]$ (see Nomenclature, p. vii). We can then write:

$$\begin{aligned} \left(\begin{bmatrix} \mathrm{M}_0 + \mathrm{G}_0 & 0 \\ 0 & -I_{n\times n} \end{bmatrix} + \begin{bmatrix} -\mathrm{K}_0 & 0 \\ 0 & 0 \end{bmatrix} \Lambda + \begin{bmatrix} \mathrm{G}_1 + \mathrm{G}_2 & \mathrm{G}_3 \\ I_{n\times n} & 0 \end{bmatrix} \tau \right) \mathbf{q}_1 = \mathbf{0} \\ \left(\begin{bmatrix} \bar{\mathrm{M}}_0 + \bar{\mathrm{G}}_0 & 0 \\ 0 & -I_{n\times n} \end{bmatrix} + \begin{bmatrix} -\bar{\mathrm{K}}_0 & 0 \\ 0 & 0 \end{bmatrix} \Lambda + \begin{bmatrix} \bar{\mathrm{G}}_1 + \bar{\mathrm{G}}_2 & \bar{\mathrm{G}}_3 \\ I_{n\times n} & 0 \end{bmatrix} \tau \right) \mathbf{q}_2 = \mathbf{0} \end{aligned} \tag{4.3.6}$$

where the upper block represents Eq. 4.3.1, and the lower block represents the identity $I_{n\times n}(\tau\mathbf{x}) = \tau I_{n\times n}\mathbf{x}$. Note that other linearisations are possible: for example,

$$\left(\begin{bmatrix} \mathrm{M}_0 + \mathrm{G}_0 & \mathrm{G}_1 + \mathrm{G}_2 \\ 0 & -I_{n\times n} \end{bmatrix} + \begin{bmatrix} -\mathrm{K}_0 & 0 \\ 0 & 0 \end{bmatrix} \Lambda + \begin{bmatrix} 0 & \mathrm{G}_3 \\ I_{n\times n} & 0 \end{bmatrix} \tau \right) \mathbf{q}_1 = \mathbf{0}. \tag{4.3.7}$$

However, in general there is no significant advantage in choosing any particular form over another. If there are large differences in the norms of the coefficient matrices (e.g. if the aeroelastic structure has high stiffness), then Higham et al. [14] have shown that these linearisations may induce large errors in the computed eigenvalues. To remedy this, a scaled linearisation is available [14,15]. However, this scaled linearisation has so far only been applied to the single-parameter eigenvalue problem, and further work is needed to generalise it to the multiparameter case. In any case, the linearisation produces a system of the form:

$$\begin{aligned} (\mathrm{A}_1 + \mathrm{B}_1\Lambda + \mathrm{C}_1\tau)\mathbf{q}_1 = \mathbf{0}, \\ (\mathrm{A}_2 + \mathrm{B}_2\Lambda + \mathrm{C}_2\tau)\mathbf{q}_2 = \mathbf{0}. \end{aligned} \tag{4.3.8}$$

If the original quadratic system has coefficient matrices of size $n \times n$, then the coefficient matrices in the linearised problem (Eq. 4.3.8) will be of size $2n \times 2n$. The operator determinants will thus be of size $4n^2 \times 4n^2$. We should note that for most aeroelastic problems it is not actually necessary to linearise the system's conjugate equation (the

second equation in Eq. 4.3.1), as when the eigenvalues are real then the second equation in Eq. 4.3.8 is the conjugate of the first ($\mathrm{A}_2 = \overline{\mathrm{A}_1}$, etc.). However, when the eigenvalues are complex and/or the second equation in Eq. 4.3.1 is not the conjugate of the first, the linearisation of each equation will need to be performed separately. In our case the second equation will inevitably be the conjugate of the first, and so in future we will only linearise one of the equations, and leave the linearisation of the second equation to be deduced from this.

This same linearisation process can be applied to any quadratic multiparameter eigenvalue problem. For the first equation in Eq. 4.3.2, which is of the form

$$(\mathrm{A} + \mathrm{B}\lambda + \mathrm{C}\tau + \mathrm{D}\lambda^2 + \mathrm{E}\tau^2)\mathbf{x} = \mathbf{0}, \tag{4.3.9}$$

we define the eigenvector $\mathbf{q} = [\mathbf{x}; \tau\mathbf{x}; \lambda\mathbf{x}]$ and obtain

$$\left(\begin{bmatrix} \mathrm{A} & \mathrm{B} & \mathrm{C} \\ 0 & -I_{n\times n} & 0 \\ 0 & 0 & -I_{n\times n} \end{bmatrix} + \begin{bmatrix} 0 & \mathrm{D} & 0 \\ I_{n\times n} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\lambda + \begin{bmatrix} 0 & 0 & \mathrm{E} \\ 0 & 0 & 0 \\ I_{n\times n} & 0 & 0 \end{bmatrix}\tau\right)\mathbf{q} = \mathbf{0}. \tag{4.3.10}$$

For the first equation in Eq. 4.3.3, which is of the form

$$(\mathrm{A} + \mathrm{B}\chi + \mathrm{C}\chi^2 + \mathrm{D}\Upsilon\chi + \mathrm{E}\Upsilon^2)\mathbf{x} = \mathbf{0} \tag{4.3.11}$$

we define the eigenvector $\mathbf{q} = [\mathbf{x}; \chi\mathbf{x}; \Upsilon\mathbf{x}]$ and obtain

$$\left(\begin{bmatrix} \mathrm{A} & \mathrm{B} & 0 \\ 0 & -I_{n\times n} & 0 \\ 0 & 0 & -I_{n\times n} \end{bmatrix} + \begin{bmatrix} 0 & \mathrm{C} & \mathrm{D} \\ I_{n\times n} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\chi + \begin{bmatrix} 0 & 0 & \mathrm{E} \\ 0 & 0 & 0 \\ I_{n\times n} & 0 & 0 \end{bmatrix}\Upsilon\right)\mathbf{q} = \mathbf{0}. \tag{4.3.12}$$

If the original quadratic system has coefficient matrices of size $n \times n$, then the coefficient matrices for both these linearised problems will be of size $3n \times 3n$. The operator determinants with thus be of size $9n^2 \times 9n^2$.

We will generally find that these linearised problems are singular. This is not universal: Eq. 4.3.6 and Eq. 4.3.7 would be nonsingular but for the fact that matrix G_3 is not full rank (cf. Eq. 4.3.4). However, Eq. 4.3.10 and Eq. 4.3.12 are singular irrespective of the form of the original coefficient matrices. Though we cannot prove rigorously that there do not exist

nonsingular two-parameter linearisations for these problems, none are known. Until recently this has been a significant obstacle to the solution of polynomial multiparameter eigenvalue problems. However, with the algorithm for the extraction of the finite regular part of GEP pencils noted in Section 4.2.4, the singular operator determinants of these linearised problems may be compressed into smaller nonsingular ones. This allows us to use the operator determinant method to solve these systems.

4.3.3 Quasi-linearisation

Hochstenbach at al. [6] recently presented another method of linearisation. Considering again Eq. 4.3.1, we can define a new eigenvalue parameter $\alpha = \tau^2$. The equations then become

$$\begin{aligned} \left((\mathrm{M}_0 + \mathrm{G}_0) + (\mathrm{G}_1 + \mathrm{G}_2)\tau + \mathrm{G}_3\alpha - \mathrm{K}_0\Lambda\right)\mathbf{x} &= \mathbf{0}, \\ \left((\overline{\mathrm{M}_0} + \overline{\mathrm{G}_0}) + (\overline{\mathrm{G}_1} + \overline{\mathrm{G}_2})\tau + \overline{\mathrm{G}_3}\alpha - \overline{\mathrm{K}_0}\Lambda\right)\bar{\mathbf{x}} &= \mathbf{0}. \end{aligned} \tag{4.3.13}$$

This is now a linear three-parameter eigenvalue problem, but with only two equations. We need a third equation to constrain the system. We have the relation $\alpha - \tau^2 = 0$, but this is nonlinear. However, noticing that we can write this relation as

$$\det\left(\begin{bmatrix} \alpha & \tau \\ \tau & 1 \end{bmatrix}\right) = 0 \tag{4.3.14}$$

we can recast it as the multiparameter eigenvalue problem

$$\left(\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\tau + \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}\alpha\right)\mathbf{y} = \mathbf{0} \tag{4.3.15}$$

for an arbitrary eigenvector $\mathbf{y}$. We thus have the linear three-parameter eigenvalue problem

$$\begin{aligned} \left((\mathrm{M}_0 + \mathrm{G}_0) + (\mathrm{G}_1 + \mathrm{G}_2)\tau + \mathrm{G}_3\alpha - \mathrm{K}_0\Lambda\right)\mathbf{x} &= \mathbf{0}, \\ \left((\overline{\mathrm{M}_0} + \overline{\mathrm{G}_0}) + (\overline{\mathrm{G}_1} + \overline{\mathrm{G}_2})\tau + \overline{\mathrm{G}_3}\alpha - \overline{\mathrm{K}_0}\Lambda\right)\bar{\mathbf{x}} &= \mathbf{0}, \\ \left(\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\tau + \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}\alpha\right)\mathbf{y} &= \mathbf{0}. \end{aligned} \tag{4.3.16}$$

In a similar way, we can linearise Eq. 4.3.2 with the definitions $\alpha = \tau^2$ and $\beta = \lambda^2$:

$$
\begin{aligned}
&\left((\mathrm{M}_0+\mathrm{G}_0)+(\mathrm{G}_1+\mathrm{G}_2)\tau+\mathrm{G}_3\alpha-\mathrm{D}_0\lambda-\mathrm{K}_0\beta\right)\mathbf{x}=\mathbf{0},\\
&\left((\overline{\mathrm{M}_0}+\overline{\mathrm{G}_0})+(\overline{\mathrm{G}_1}+\overline{\mathrm{G}_2})\tau+\overline{\mathrm{G}_3}\alpha-\overline{\mathrm{D}_0}\lambda-\overline{\mathrm{K}_0}\beta\right)\bar{\mathbf{x}}=\mathbf{0},\\
&\left(\begin{bmatrix}0&0\\0&1\end{bmatrix}+\begin{bmatrix}0&1\\1&0\end{bmatrix}\tau+\begin{bmatrix}1&0\\0&0\end{bmatrix}\alpha\right)\mathbf{y_1}=\mathbf{0},\\
&\left(\begin{bmatrix}0&0\\0&1\end{bmatrix}+\begin{bmatrix}0&1\\1&0\end{bmatrix}\lambda+\begin{bmatrix}1&0\\0&0\end{bmatrix}\beta\right)\mathbf{y_2}=\mathbf{0},
\end{aligned}
\tag{4.3.17}
$$

and Eq. 4.3.3 with the definitions $\alpha=\chi^2$, $\beta=\Upsilon^2$, $\gamma=\Upsilon\chi$:

$$
\begin{aligned}
&\left((\mathrm{M}_0\ +\mathrm{G}_0)\alpha+(\mathrm{G}_1+\mathrm{G}_2)\gamma+\mathrm{G}_3\beta-\mathrm{D}_0\chi-\mathrm{K}_0\right)\mathbf{x}=\mathbf{0},\\
&\left((\overline{\mathrm{M}_0}+\overline{\mathrm{G}_0})\alpha+(\overline{\mathrm{G}_1}+\overline{\mathrm{G}_2})\gamma+\overline{\mathrm{G}_3}\beta-\overline{\mathrm{D}_0}\chi-\overline{\mathrm{K}_0}\right)\mathbf{x}=\mathbf{0},\\
&\left(\begin{bmatrix}0&0\\0&1\end{bmatrix}+\begin{bmatrix}0&1\\1&0\end{bmatrix}\chi+\begin{bmatrix}1&0\\0&0\end{bmatrix}\alpha\right)\mathbf{y_1}=\mathbf{0},\\
&\left(\begin{bmatrix}0&1\\0&0\end{bmatrix}\alpha+\begin{bmatrix}1&0\\0&1\end{bmatrix}\gamma+\begin{bmatrix}0&0\\1&0\end{bmatrix}\beta\right)\mathbf{y_2}=\mathbf{0},
\end{aligned}
\tag{4.3.18}
$$

These systems can be solved via the operator determinant method, in a more general form than that presented in Section 4.2.4.

4.3.4 The general operator determinant method

The general form of an inhomogeneous multiparameter eigenvalue problem is

$$
W_i(\boldsymbol{\eta})=\mathrm{A}_{i0}\mathbf{x}_i+\sum_{j=1}^{N}\eta_j\mathrm{A}_{ij}\mathbf{x}_i\,,\qquad i=1,\dots,n \tag{4.3.19}
$$

where $\boldsymbol{\eta}$ is the vector of eigenvalues (η_j being the individual eigenvalues), A_{ij} are the coefficient matrices, which can be complex and of different sizes for each equation, and $\mathbf{x}_i$ are the eigenvectors. Note that A_{ij} refers to the i,j-th matrix coefficient, not to the elements of a single matrix A. Though we will not attempt to derive them, it is possible to define operator determinants for Eq. 4.3.19 that perform a function analogous to the operator determinants in the two-parameter case. There are $N+1$ such operator determinants (Δ_0 through to Δ_N). They can be defined as follows [16–18]:

$$\Delta_0 = \begin{vmatrix} \mathrm{A}_{11} & \mathrm{A}_{12} & \cdots & \mathrm{A}_{1N} \\ \mathrm{A}_{21} & \mathrm{A}_{22} & \cdots & \mathrm{A}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{A}_{N1} & \mathrm{A}_{N2} & \cdots & \mathrm{A}_{NN} \end{vmatrix}_{\otimes} \tag{4.3.20}$$

and

$$\Delta_1 = \begin{vmatrix} -\mathrm{A}_{10} & \mathrm{A}_{12} & \cdots & \mathrm{A}_{1N} \\ -\mathrm{A}_{20} & \mathrm{A}_{22} & \cdots & \mathrm{A}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ -\mathrm{A}_{n0} & \mathrm{A}_{N2} & \cdots & \mathrm{A}_{NN} \end{vmatrix}_{\otimes}, \tag{4.3.21}$$

$$\Delta_i = \begin{vmatrix} \mathrm{A}_{11} & \cdots & \mathrm{A}_{1,i-1} & -\mathrm{A}_{10} & \mathrm{A}_{1,i+1} & \cdots & \mathrm{A}_{1N} \\ \mathrm{A}_{21} & \cdots & \mathrm{A}_{2,i-1} & -\mathrm{A}_{20} & \mathrm{A}_{2,i+1} & \cdots & \mathrm{A}_{2N} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ \mathrm{A}_{N1} & \cdots & \mathrm{A}_{N,i-1} & -\mathrm{A}_{N0} & \mathrm{A}_{N,i+1} & \cdots & \mathrm{A}_{NN} \end{vmatrix}_{\otimes}, \tag{4.3.22}$$

$$\Delta_N = \begin{vmatrix} \mathrm{A}_{11} & \cdots & \mathrm{A}_{1,N-1} & -\mathrm{A}_{10} \\ \mathrm{A}_{21} & \cdots & \mathrm{A}_{2,N-1} & -\mathrm{A}_{20} \\ \vdots & & \vdots & \vdots \\ \mathrm{A}_{N1} & \cdots & \mathrm{A}_{N,N-1} & -\mathrm{A}_{N0} \end{vmatrix}_{\otimes}, \tag{4.3.23}$$

where $|\cdot|_{\otimes}$ represents a determinant operation in which the multiplication operations that are used to relate elements in a normal determinant computation are replaced by Kronecker products. In the future we will simply notate this with usual determinant brackets, $|\cdot|$, as the meaning will be clear from the context. Note that it is possible to construct and define these determinants in a more rigorous way than we have done [16–21]; however, this would require us to go through a lot of underlying theory which is not otherwise relevant. Definitions equivalent to those of Eq. 4.3.20 – 4.3.23 are found in [16,17] and implemented in [8].

With these operator determinants, we can then compute the eigenvalues. Providing Δ_0 is nonsingular, it holds that

$$\Delta_i \mathbf{x} = \eta_i \Delta_0 \mathbf{x} \tag{4.3.24}$$

which, as a generalised eigenvalue problem, can be solved by well-known methods such as the QZ algorithm [13]. No algorithms have yet been developed for the compression (i.e. extraction of the finite regular part) of these generalised eigenvalue problems when Δ_0 is singular and $N > 2$. It may be straightforward to extend the $N = 2$ algorithm presented in [3]: this needs further investigation. Applying the general operator determinant method to

Eq. 4.3.16 results in operator determinants of size $2n^2 \times 2n^2$, for original coefficient matrices of size $n \times n$. Applying the method to Eq. 4.3.17 or Eq. 4.3.18 results in operator determinants of size $4n^2 \times 4n^2$. These are both smaller than the corresponding operator determinants when the standard linearisation is applied (Section 4.3.2).

It is worthwhile presenting the details of the definition and computation of these general operator determinants. Firstly, as regards their definition. Consider a general system with three eigenvalue variables and three equations (e.g. Eq. 4.3.16):

$$\begin{aligned}(\mathrm{A}_{10} + \mathrm{A}_{11}\eta_1 + \mathrm{A}_{12}\eta_2 + \mathrm{A}_{13}\eta_3)x_1 &= 0,\\ (\mathrm{A}_{20} + \mathrm{A}_{21}\eta_1 + \mathrm{A}_{22}\eta_2 + \mathrm{A}_{23}\eta_3)x_2 &= 0,\\ (\mathrm{A}_{30} + \mathrm{A}_{31}\eta_1 + \mathrm{A}_{32}\eta_2 + \mathrm{A}_{33}\eta_3)x_3 &= 0.\end{aligned} \tag{4.3.25}$$

The most instructive way to visualise these coefficients is in the form of a 3×4 matrix:

$$\begin{bmatrix}\mathrm{A}_{10} & \mathrm{A}_{11} & \mathrm{A}_{12} & \mathrm{A}_{13}\\ \mathrm{A}_{20} & \mathrm{A}_{21} & \mathrm{A}_{22} & \mathrm{A}_{23}\\ \mathrm{A}_{30} & \mathrm{A}_{31} & \mathrm{A}_{32} & \mathrm{A}_{33}\end{bmatrix}. \tag{4.3.26}$$

This matrix is not square and so we cannot define a meaningful operator determinant for it. The operator determinant matrices are formed by removing and rearranging columns. Δ_0 is formed by simply removing the first row (A_{i0}).

$$\Delta_0 = \begin{vmatrix}\mathrm{A}_{11} & \mathrm{A}_{12} & \mathrm{A}_{13}\\ \mathrm{A}_{21} & \mathrm{A}_{22} & \mathrm{A}_{23}\\ \mathrm{A}_{31} & \mathrm{A}_{32} & \mathrm{A}_{33}\end{vmatrix} \tag{4.3.27}$$

and the other determinants are formed by removing the first row and then also removing the i-th row (for Δ_i) and then replacing it with the negative of the original first row ($-\mathrm{A}_{k0}$)

$$\Delta_1 = \begin{vmatrix}-\mathrm{A}_{10} & \mathrm{A}_{12} & \mathrm{A}_{13}\\ -\mathrm{A}_{20} & \mathrm{A}_{22} & \mathrm{A}_{23}\\ -\mathrm{A}_{30} & \mathrm{A}_{32} & \mathrm{A}_{33}\end{vmatrix}, \tag{4.3.28}$$

$$\Delta_2 = \begin{vmatrix}\mathrm{A}_{11} & -\mathrm{A}_{10} & \mathrm{A}_{13}\\ \mathrm{A}_{21} & -\mathrm{A}_{20} & \mathrm{A}_{23}\\ \mathrm{A}_{31} & -\mathrm{A}_{30} & \mathrm{A}_{33}\end{vmatrix}, \tag{4.3.29}$$

$$\Delta_3 = \begin{vmatrix}\mathrm{A}_{11} & \mathrm{A}_{12} & -\mathrm{A}_{10}\\ \mathrm{A}_{21} & \mathrm{A}_{22} & -\mathrm{A}_{20}\\ \mathrm{A}_{31} & \mathrm{A}_{32} & -\mathrm{A}_{30}\end{vmatrix}. \tag{4.3.30}$$

In the two-parameter case these operator equations collapse into the well-known Eq. 4.2.10, because there exist simple expressions for the determinants. So for a system of the form

$$\begin{aligned}(\mathrm{A}_1 + \mathrm{B}_1\eta_1 + \mathrm{C}_1\eta_2)x_1 = 0,\\ (\mathrm{A}_2 + \mathrm{B}_2\eta_1 + \mathrm{C}_2\eta_2)x_2 = 0.\end{aligned} \tag{4.3.31}$$

we have

$$\begin{aligned}\Delta_0 &= \begin{vmatrix}\mathrm{B}_1 & \mathrm{C}_1\\ \mathrm{B}_2 & \mathrm{C}_2\end{vmatrix} = \mathrm{B}_1 \otimes \mathrm{C}_2 - \mathrm{C}_1 \otimes \mathrm{B}_2\\ \Delta_1 &= \begin{vmatrix}-\mathrm{A}_1 & \mathrm{C}_1\\ -\mathrm{A}_2 & \mathrm{C}_2\end{vmatrix} = \mathrm{C}_1 \otimes \mathrm{A}_2 - \mathrm{A}_1 \otimes \mathrm{C}_2\\ \Delta_2 &= \begin{vmatrix}\mathrm{B}_1 & -\mathrm{A}_1\\ \mathrm{B}_2 & -\mathrm{A}_2\end{vmatrix} = \mathrm{A}_1 \otimes \mathrm{B}_2 - \mathrm{B}_1 \otimes \mathrm{A}_2\end{aligned} \tag{4.3.32}$$

Which agrees exactly with the two-parameter operator determinants as given in Eq. 4.2.10 (Section 4.2.3).

4.3.5 Computing operator determinants

In the two-parameter case, the computation of the operator determinants is trivial. However in the case of larger systems we must find some other approach. In the general case, the operator determinants may be computed by modifying the Leibniz formula for the determinant [4,16]:

$$\Delta(\mathrm{M}) = \sum_{\mathbf{s} \in S_N} \mathrm{sgn}(\mathbf{s}) \bigotimes_{i=1}^{N} \mathrm{M}_{i,\mathbf{s}_i} \tag{4.3.33}$$

where S_N is the set of permutations of the set $\{1, \ldots, N\}$, $\mathrm{sgn}(\mathbf{s})$ is the sign of the permutation vector $\mathbf{s} \in S_N$. The notation $\bigotimes_i^N$ denotes the repeated application of the Kronecker product, in the same way that Σ_i^N denotes repeated summation and Π_i^N repeated multiplication. Note that the tensor determinant definition in [16] is slightly erroneous as the factor $(-1)^{\mathrm{sgn}(\sigma)}$ in their tensor determinant expressions should be either $\mathrm{sgn}(\sigma)$ or $(-1)^{N(\sigma)}$, where $N(\sigma)$ is the number of inversions in σ. While this thesis was being written, Muhič and Plestenjak [8] published a general algorithm based on the Laplace expansion [22]

that is able to compute the operator determinants of a system of arbitrary size, via a process of recursion:

$$\Delta(\mathrm{M}) = \sum_{i=1}^{N} (-1)^{i+1} \mathrm{M}_{i1} m_{i1}, \tag{4.3.34}$$

where m_{i1} denotes the minor entry corresponding to M_{i1}. The summation in Eq. 4.3.34 follows the first column of M, though, of course, many other summation paths could be used. It should be noted that the use of the Laplace or Leibniz methods for computing the determinant of an ordinary matrix have very high non-polynomial computational complexity costs – $\mathcal{O}(n!)$ and $\mathcal{O}(n!\,n)$, respectively [23,24]. As of yet no other methods have been developed for computing operator determinants, however, it is may be possible possible that a Gaussian elimination / LU decomposition method could be developed for multiparameter systems. Such a method would have lower computational complexity – for Gaussian elimination on standard matrices the cost is $\mathcal{O}(n^3)$. It could also allow us to devise entirely new ways of solving the operator determinant GEP. For example reducing the multiparameter system to a row-echelon form, compute the determinant by multiplying along the diagonal, and making use of the Kronecker product spectrum result [4]. Alternatively, we could reduce the system to reduced row-echelon form, i.e. a series of one-parameter problems which can easily be solved with conventional methods. Either of these methods would be completely novel, however at the current time there is no assurance that they could be successfully developed.

4.3.6 Numerical experiments

To validate the algorithms and methods we have so far presented, we solve each of the three quadratic systems noted in Section 4.3.1 for their flutter points, taking parameter values from Chapter 2. Consider first the section model with quasisteady aerodynamics and no structural damping, Eq. 4.3.1. Figure 4.5 shows a contour plot of the system. As described in Chapter 3, the physical flutter point is located at $\tau_F = 0.996$, $\Lambda_F = 0.568\ s^2/\mathrm{rad}^2$. One non-physical flutter point can be located at $\tau_F = -0.996$, and two other points of neutral stability at $\tau = 0$ and $\Lambda = 3.33\ s^2/\mathrm{rad}^2$, $\Lambda = 0.492\ s^2/\mathrm{rad}^2$. The divergence point of this system occurs at infinite τ and Λ and so cannot be located by the contour plot. It has been shown in Chapter 3 that the contour plot for this system agrees

with its modal damping plot. Figure 4.5 also shows the results of the direct solver with compression (Algorithm 3.2) applied to the linearisation of Eq. 4.3.1 (i.e. Eq. 4.3.6) .As can be seen, the results agree exactly – the direct solver locates the flutter points at $\tau_F = \pm 0.9963$ Hz, $\Lambda_F = 0.5675\ s^2/\mathrm{rad}^2$ and the points of zero-airspeed neutral stability at $\Lambda = 3.327\ s^2/\mathrm{rad}^2$, $\Lambda = 0.4916\ s^2/\mathrm{rad}^2$. The computation time for the direct solvers is too small for any meaningful computational assessment to be performed – we would need to devise a larger system.

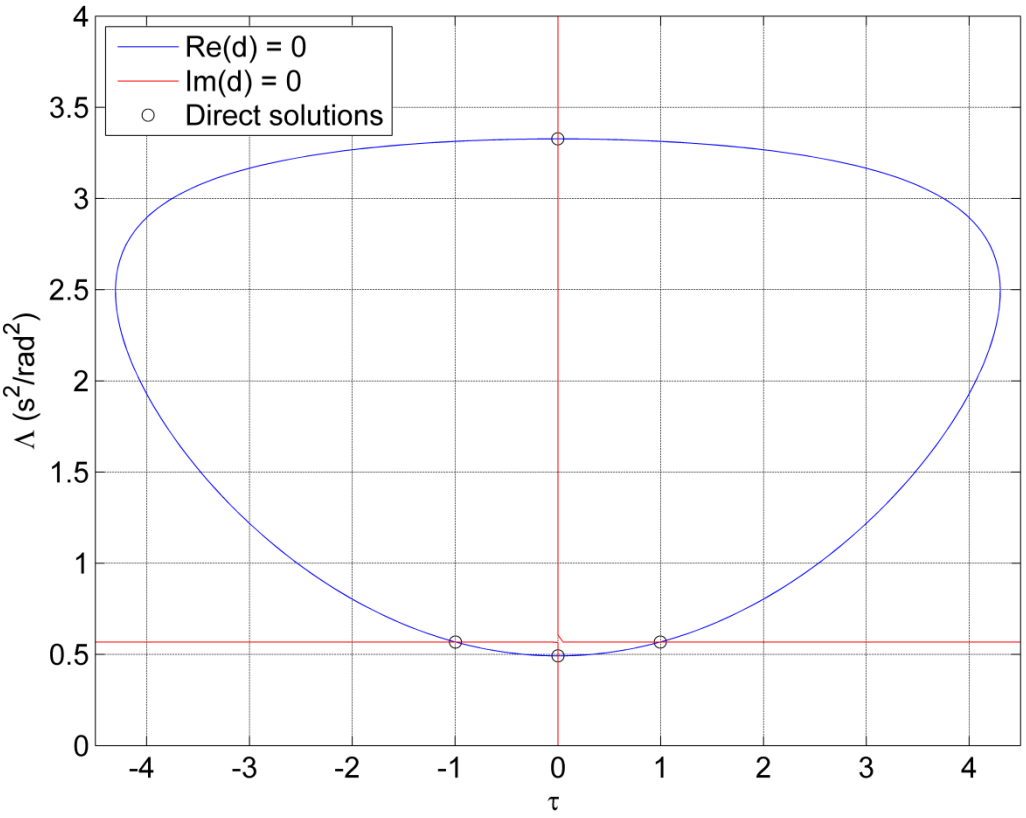


Figure 4.5: Contour plot for the section model with quasisteady Theodorsen aerodynamics and no structural damping, showing solutions from direct solver (linearisation and quasi-linearisation).

Applying the quasi-linearisation to this problem yields a singular system. This is only because the matrix G_3 is singular – in the general case where the coefficient matrices are nonsingular, the linearised problem is nonsingular. We do not have a compression algorithm (nor indeed, any extension of the common regular part relationship noted in Section 4.2.4) that applies to the general singular multiparameter problem. However, though it only takes Δ_0, Δ_1 and Δ_2 as an input, it will successfully compress these three operator determinants

for the general multiparameter problem (ignoring the others). In the case of Eq. 4.3.16 (and indeed also Eq. 4.3.17 and Eq. 4.3.18), with these three operator determinants we can solve for all the eigenvalue parameters of the system, irrespective of the order in which we arrange the eigenvalues. However, in the case where we have a higher-dimensional singular multiparameter system which has eigenvalues which are not related as are those of Eq. 4.3.16, it may be more difficult to extract the eigenvalues not defined by GEPs in Δ_0, Δ_1 and Δ_2.

However, in any case, we have absolutely no justification for using the two-parameter compression algorithm to compress higher-dimension problems other than evidence that the algorithm actually does work. The paper by Muhič and Plestenjak [3] only proved the relationship between the common regular part of the singular GEP and the eigenvalues of the multiparameter problem for two-parameter problems arising from linearisation, and the compression algorithm was only derived for this situation. It is not even clear how to define a 'common regular part' when more than two pencils are involved – would it refer to the regular part that is common to all the pencils, or will the common regular part have to be defined between successive pairs of pencils? The fact that the two-parameter algorithm does in fact work when applied to the first three operator determinants of the general singular problem suggests that the theorem will generalise; it is merely a question of working out what the generalisation is. This is a very interesting area for further research. However, for the moment we need not worry about these problems, as the fact that the compression algorithm does work allows us to continue with the solution of the GEP as normal. For the purposes of this thesis we use the algorithm in this way without proof.

The results are unsurprising: the computed eigenvalues are identical to those of the two-parameter linearisation, shown in Figure 4.5. This validates our quasi-linearisation and general operator determinant method. The uncompressed operator determinants are of size 8×8, and the compression operator determinants are of size 4×4. This is better than the standard linearisation, which produces uncompressed operator determinants of size 16×16 [2] and thus requires more computational effort during the compression process. We

[2] And, of course, compressed operator determinants of size 4×4, as should be self-evident from the uniqueness of the eigenvalues of the original multiparameter problem.

will see when simulating larger systems in Section 4.4.5 that the compression process occupies the majority of the computational effort expended in solving singular problems, and so this size difference could be significant.

We also simulate the section model with quasisteady Theodorsen aerodynamics and structural damping. Figure 4.6 shows a contour plot for the τ-λ form of this system (Eq. 4.3.2), alongside the solution of system as determined by Algorithm 3.2 applied to Eq. 4.3.10. Note the similarity to Figure 4.5 – though Figure 4.6 uses a y-axis dimension of λ and not Λ. As can be seen, the flutter points from the contour plot and the direct solver agree exactly. The physical flutter point can be located at $\tau_F = 1.650,\ \lambda_F = 0.8336\ \mathrm{s/rad}$. Another non-physical flutter point occurs at a small negative τ, as noted in Chapter 3. Note again that this plot does not show the divergence point: this occurs at $\chi = 0$ and thus infinite τ and λ. This may actually be useful, because it will prevent the iterative methods that we will develop in the next two chapters from converging to the divergence point rather than the flutter point.

Figure 4.7 shows a contour plot for the Υ-χ form (Eq. 4.3.3). According to the contour plot, the flutter point is located at $\Upsilon_F = 1.98\ \mathrm{Hz}$, $\chi_F = 1.20\ \mathrm{rad/s}$ and the divergence point at $\Upsilon_D = 3.99\ \mathrm{Hz}$. Figure 4.7 also shows the results of the direct solver with compression (Algorithm 3.2) applied to Eq. 4.3.12. As can be seen the results agree exactly – the direct solver locates the flutter point at $\Upsilon_F = 1.979\ \mathrm{Hz}$, $\chi_F = 1.200\ \mathrm{rad/s}$ and the divergence point at $\Upsilon_D = 3.989\ \mathrm{Hz}$. It is easy to show that this agrees well with the solution from Figure 4.5: $\lambda_F = 1/\chi_F = 0.8333\ \mathrm{s/rad}$ and $\tau_F = \Upsilon_F/\lambda_F = 1.649$. Again, the quasi-linearisation with general operator determinant method and two-parameter compression produces exactly the same results.

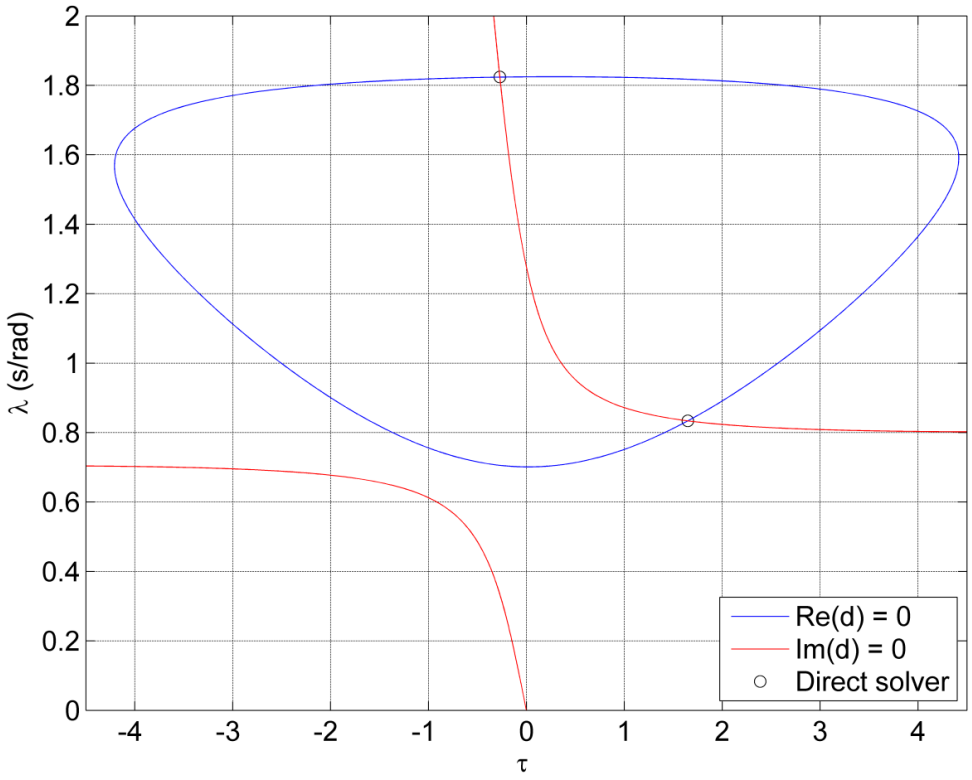


Figure 4.6: Contour plot for the section model with quasisteady Theodorsen aerodynamics

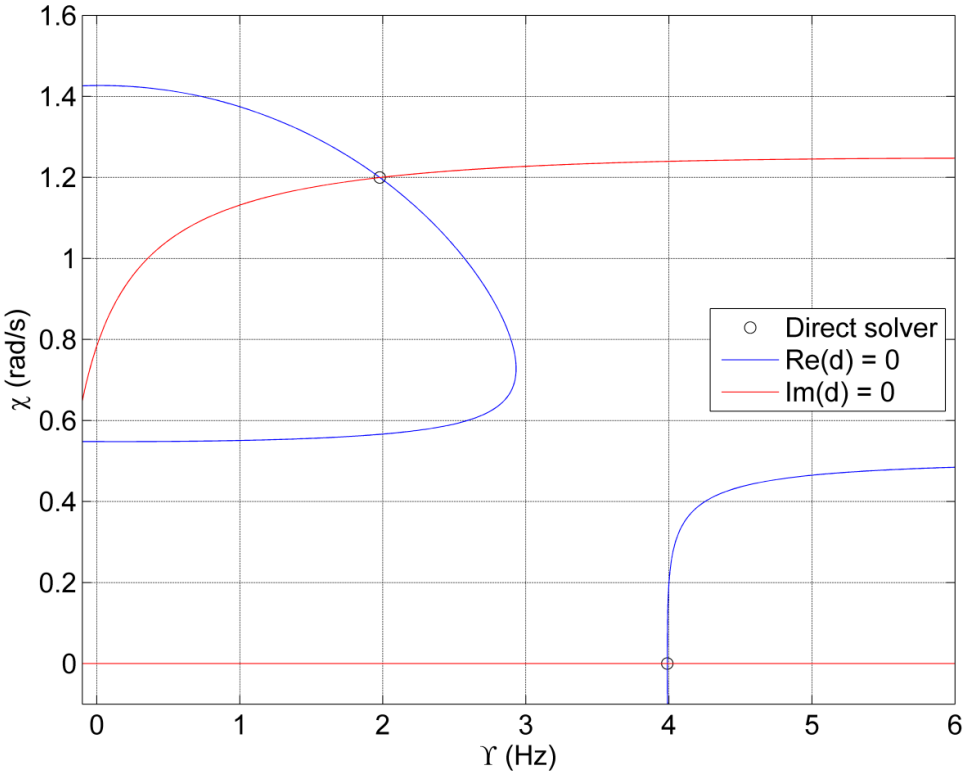


Figure 4.7: Contour plot for Quasisteady Theodorsen section model (Υ-χ form), showing solutions from direct solver.

4.4 PROBLEMS OF HIGHER ORDER

4.4.1 Motivation

In Chapter 2 we introduced two polynomial multiparameter eigenvalue problems with order greater than two, both of them connected with the approximation of Theodorsen's function by more manageable expressions. The use of Swiney's fractional-order approximation [25] yielded a problem of order 17 in $\hat{\kappa}$ and order 2 in χ, and the use of Jones' rational approximation [26] yielded a problem of order 4 in κ and order 2 in χ. We will consider these problems each in turn.

4.4.2 The Jones approximation

The use of Jones' rational approximation transforms the semi-structured problem of a section model under Theodorsen's unsteady aerodynamics to the polynomial problem

$$\begin{aligned}(\kappa^4\chi^2\mathrm{A}_0 + \kappa^3\chi^2\mathrm{A}_1 + \kappa^2\chi^2\mathrm{A}_2 + \kappa\chi^2\mathrm{A}_3 + \chi^2\mathrm{A}_4 + \kappa^4\chi\mathrm{A}_5 + \kappa^3\chi\mathrm{A}_6 + \kappa^2\chi\mathrm{A}_7 \\ + \kappa^4\mathrm{A}_8 + \kappa^3\mathrm{A}_9 + \kappa^2\mathrm{A}_{10})\mathbf{x} = \mathbf{0}.\end{aligned} \tag{4.4.1}$$

We define the eigenvector

$$\mathbf{q} = [\kappa^4\chi\mathbf{x};\ \kappa^3\chi\mathbf{x};\ \kappa^2\chi\mathbf{x};\ \kappa\chi\mathbf{x};\ \chi\mathbf{x};\ \kappa^4\mathbf{x};\ \kappa^3\mathbf{x};\ \kappa^2\mathbf{x};\ \kappa\mathbf{x};\ \mathbf{x}] \tag{4.4.2}$$

which, with much careful arrangement, allows us to rewrite the equation as

$$\left(\mathrm{M}_0 + \mathrm{M}_\chi\chi + \mathrm{M}_\kappa\kappa\right)\mathbf{q} = \mathbf{0} \tag{4.4.3}$$

with

$$\mathrm{M}_0 = \begin{bmatrix} \mathrm{A}_5 & \mathrm{A}_6 & \mathrm{A}_7 & 0 & 0 & \mathrm{A}_8 & \mathrm{A}_9 & \mathrm{A}_{10} & 0 & 0 \\ \mathrm{I} & & & & & & & & & \\ & \mathrm{I} & & & & & & & & \\ & & \mathrm{I} & & & & & & & \\ & & & \mathrm{I} & & & & & & \\ & & & & \mathrm{I} & & & & & \\ & & & & & \mathrm{I} & & & & \\ & & & & & & \mathrm{I} & & & \\ & & & & & & & \mathrm{I} & & \\ & & & & & & & & \mathrm{I} & 0 \end{bmatrix} \tag{4.4.4}$$

$$\mathrm{M}_{\chi} = \begin{bmatrix} \mathrm{A}_0 & \mathrm{A}_1 & \mathrm{A}_2 & \mathrm{A}_3 & \mathrm{A}_4 & 0 & 0 & 0 & 0 & 0 \\ & 0 & & & & & & & & \\ & & 0 & & & & & & & \\ & & & 0 & & & & & & \\ & & & & 0 & & & & & \\ & & & & & 0 & & & & -\mathrm{I} \\ & & & & & & 0 & & & \\ & & & & & & & 0 & & \\ & & & & & & & & 0 & \\ & & & & & & & & & 0 \end{bmatrix} \tag{4.4.5}$$

$$\mathrm{M}_{\kappa} = \begin{bmatrix} 0 & & & & & & & & & \\ & -\mathrm{I} & & & & & & & & \\ & & -\mathrm{I} & & & & & & & \\ & & & -\mathrm{I} & & & & & & \\ & & & & -\mathrm{I} & & & & & \\ & & & & & 0 & & & & \\ & & & & & & -\mathrm{I} & & & \\ & & & & & & & -\mathrm{I} & & \\ & & & & & & & & -\mathrm{I} & \\ & & & & & & & & & -\mathrm{I} \end{bmatrix} \tag{4.4.6}$$

where $\mathrm{I} = I_{n\times n}$ and $0 = 0_{n\times n}$ for readability. This is a linear problem which can be solved with the two-parameter operator determinant method presented in Section 4.2.3. The conjugate equation can be constructed by conjugating all the M-matrices or by conjugating their individual entries. The linearised system is of size $10n \times 10n$. We note that no linearisation of this size has ever been presented in the literature, though the method is a simple extension of the linearisation of lower-order polynomials. This linearisation may seem impractically large, but we will in fact show later that the solution times are not unreasonable.

4.4.3 The fractional-order approximation

The use of Jones' rational approximation transforms the semi-structured problem of a section model under Theodorsen's unsteady aerodynamics to the polynomial problem

$$\begin{aligned}(\hat{\kappa}^{17}\chi^2\mathrm{A}_0 + \hat{\kappa}^{12}\chi^2\mathrm{A}_1 + \hat{\kappa}^{11}\chi^2\mathrm{A}_2 + \hat{\kappa}^{6}\chi^2\mathrm{A}_3 + \hat{\kappa}^{5}\chi^2\mathrm{A}_4 + \chi^2\mathrm{A}_5 + \hat{\kappa}^{17}\chi\mathrm{A}_6 \\ + \hat{\kappa}^{12}\chi\mathrm{A}_7 + \hat{\kappa}^{17}\mathrm{A}_8 + \hat{\kappa}^{12}\mathrm{A}_9)\mathbf{x} = \mathbf{0}\end{aligned} \tag{4.4.7}$$

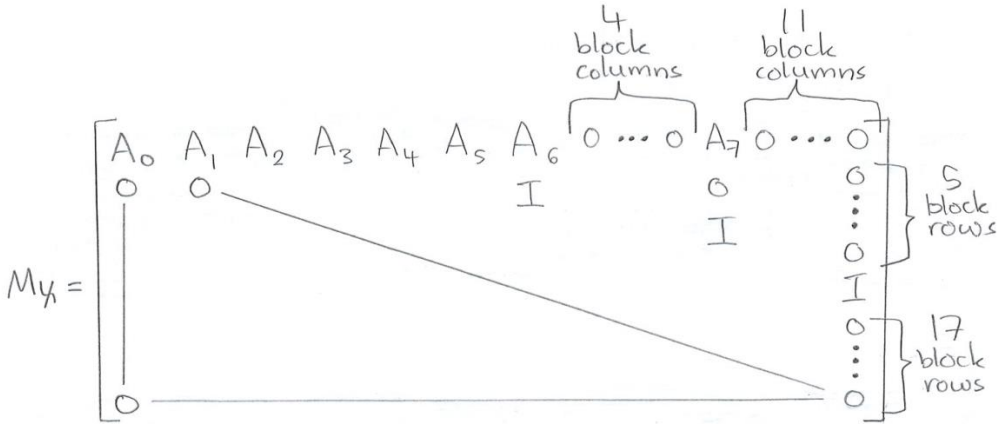
With a new eigenvalue variable $\hat{\kappa} = \sqrt[6]{\kappa}$. We define the eigenvector

$$\mathbf{q} = \hat{\kappa}^{17}\chi\mathbf{x};\ \hat{\kappa}^{12}\chi\mathbf{x};\ \hat{\kappa}^{11}\chi\mathbf{x};\ \hat{\kappa}^{6}\chi\mathbf{x};\ \hat{\kappa}^{5}\chi\mathbf{x};\ \chi\mathbf{x};$$
$$\hat{\kappa}^{17}\mathbf{x};\ \hat{\kappa}^{16}\mathbf{x};\ \hat{\kappa}^{15}\mathbf{x};\ \hat{\kappa}^{14}\mathbf{x};\ \hat{\kappa}^{13}\mathbf{x};\ \hat{\kappa}^{12}\mathbf{x};\ \hat{\kappa}^{11}\mathbf{x};\ \hat{\kappa}^{10}\mathbf{x};\ \hat{\kappa}^{9}\mathbf{x};\ \hat{\kappa}^{8}\mathbf{x}; \quad (4.4.8)$$
$$\hat{\kappa}^{7}\mathbf{x};\ \hat{\kappa}^{6}\mathbf{x};\ \hat{\kappa}^{5}\mathbf{x};\ \hat{\kappa}^{4}\mathbf{x};\ \hat{\kappa}^{3}\mathbf{x};\ \hat{\kappa}^{2}\mathbf{x};\ \hat{\kappa}\mathbf{x};\ \mathbf{x};$$

which allows us to write the equation as

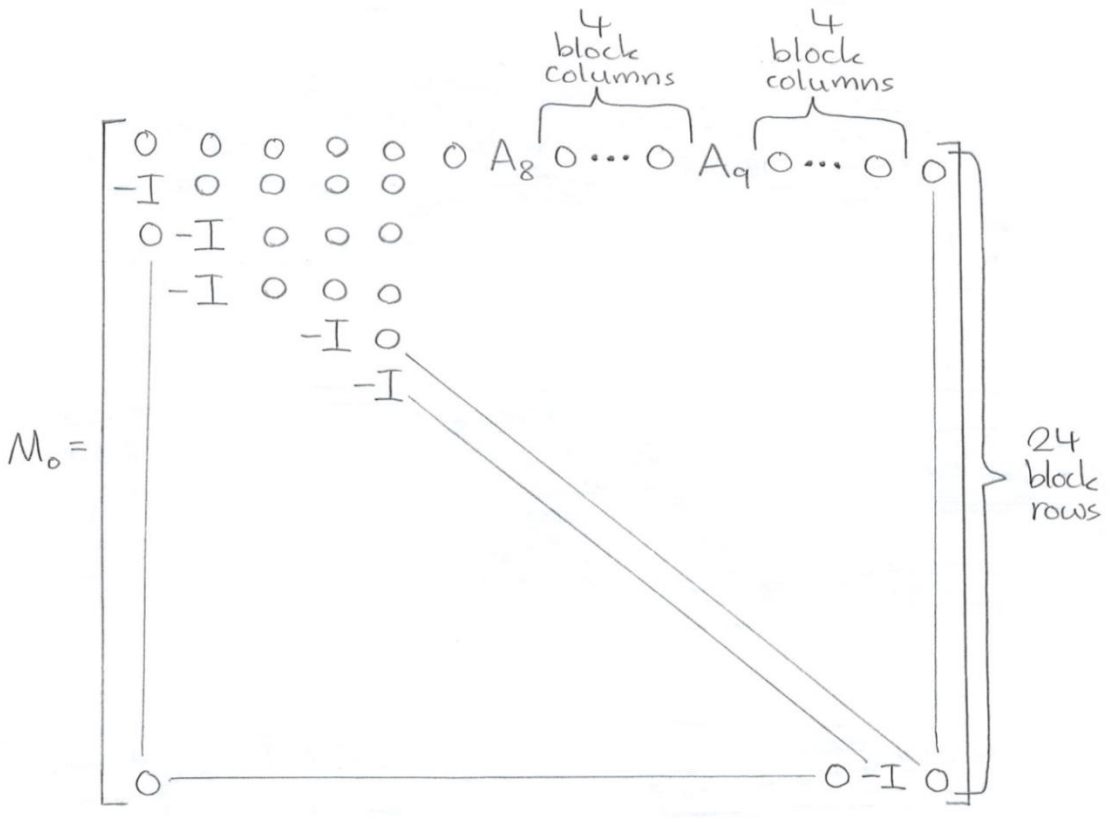
$$\left(\mathrm{M}_0 + \mathrm{M}_\chi\chi + \mathrm{M}_{\hat{\kappa}}\hat{\kappa}\right)\mathbf{q} = \mathbf{0} \quad (4.4.9)$$

with

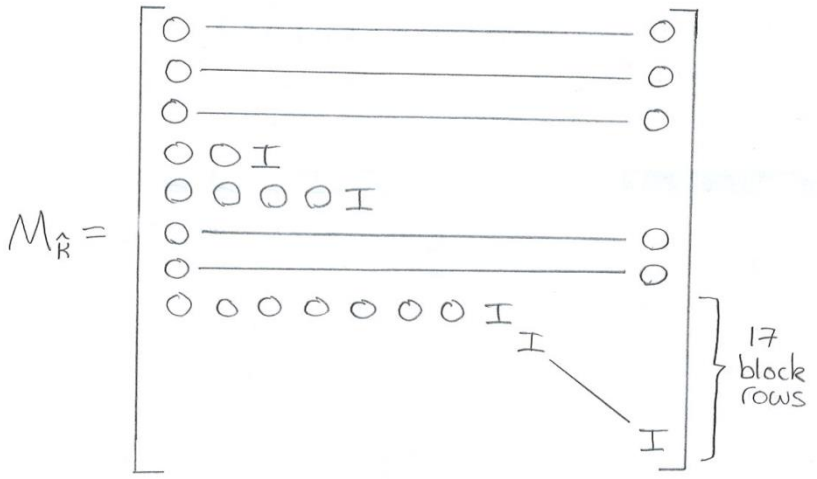


(4.4.10)

and



(4.4.11)



(4.4.12)

where $\mathrm{I} = I_{n\times n}$ and $0 = 0_{n\times n}$ for readability. We revert to writing these definitions by hand due to the inability of a widely-used word processor. This is again a linear system, of size $24n \times 24n$, that can be solved by the two-parameter operator determinant method (Section 4.2.3).

4.4.4 Optimality of linearisations

When we are dealing with systems as large as these, the question of whether we can devise a linearisation that still accurately represents the system but is of smaller size becomes apparent. It is well known that the linearisations presented earlier in this chapter are not optimal: from the theory on determinantal representations [27], there exist matrices M_0, M_χ and M_κ of size $2n \times 2n$ such that [3]

$$\det\left(\mathrm{M}_0 + \mathrm{M}_\chi\chi + \mathrm{M}_\kappa\kappa\right) = \det(\mathrm{A} + \mathrm{B}\chi + \mathrm{C}\kappa + \mathrm{D}\chi\kappa + \mathrm{E}\chi^2 + \mathrm{F}\kappa^2). \qquad (4.4.13)$$

It follows that the operator determinants for such a system are of size $4n^2 \times 4n^2$. Such a linearisation would be both nonsingular and smaller than those previously presented; however, no method for constructing such a linearisation is currently known. Eq. 4.4.13 remains a purely theoretical result. In this context it is difficult to assess whether the linearisations presented earlier could be made smaller. When using operator determinants to solve the linearised system, the method of quasi-linearisation (Section 4.3.3) is more efficient than the standard linearisation: Eq. 4.3.16, 4.3.17 and 4.3.18 produce operator determinants of size $2n^2 \times 2n^2$, $4n^2 \times 4n^2$ and $4n^2 \times 4n^2$ respectively. The standard

linearisations of these systems produce determinants of size $4n^2 \times 4n^2$, $9n^2 \times 9n^2$ and $9n^2 \times 9n^2$. These sizes cannot be compared directly with the optimal operator determinant size ($4n^2 \times 4n^2$) as Eq. 4.4.13 contains more terms than are present in any of the equations we deal with. However, if we were to linearise such a system, we would obtain linearised matrices of size $3n \times 3n$ (thus operator determinants of size $9n^2 \times 9n^2$), and quasi-linearised operator determinants of size $8n^2 \times 8n^2$ [3,6].

4.4.5 Numerical experiments

To begin our numerical study of these systems with approximate Theodorsen aerodynamics, we might be interested in the accuracy of the approximations to Theodorsen's function that they employ. Figures 4.8 and 4.9 show the comparison between Theodorsen's function and these approximations, in terms of real and imaginary parts. The maximum errors in the real part of both approximations are not very significant, being on the order of 2%. The error in the imaginary part of the fractional approximation is of similar order; however, the maximum error in the imaginary part of Jones' approximation is slightly more significant at 8%). Though the maximum absolute error in Jones' approximation is one-and-a-half times larger than the maximum absolute error in the fractional approximation, both errors are small (0.015 and 0.010) and in terms of percentage error work out to be roughly equivalent at 2%. The fractional approximation should thus be considered only marginally more accurate than the Jones' model, and it requires a much larger linearised system (Section 4.4.2 and 4.4.3). Both approximations are quite suitable for industrial simulation, where the inaccuracy of Theodorsen's aerodynamic theory in modelling the system of interest is likely to far outweigh the accuracies of these approximations to that theory.

Figures 4.10 and 4.11 show the direct flutter-point solutions for the fractional approximation, superimposed over contour plots of the exact Theodorsen system. Figure 4.10 shows the solutions in terms of χ and $\hat{\kappa}$ (the variables computed directly by the solver), and Figure 4.11 shows them in terms of χ and $\kappa = \hat{\kappa}^6$. As can be seen in both figures, the flutter points computed by the direct solver are not visually different to those that may be seen on the contour plot. The exact system has flutter points at $\kappa = 5.4 \times 10^{-4}$ and $\kappa = 0.283$, and the flutter points from the direct solver are identical to these values at the stated precision.

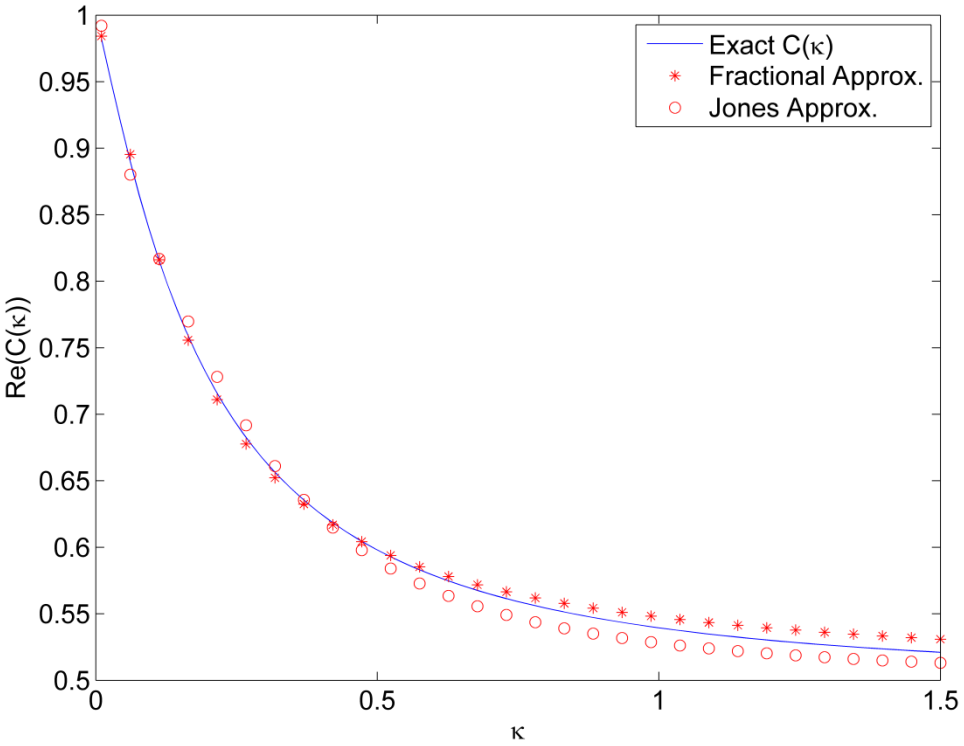


Figure 4.8: Comparison between the real part of Theodorsen's function and its approximations

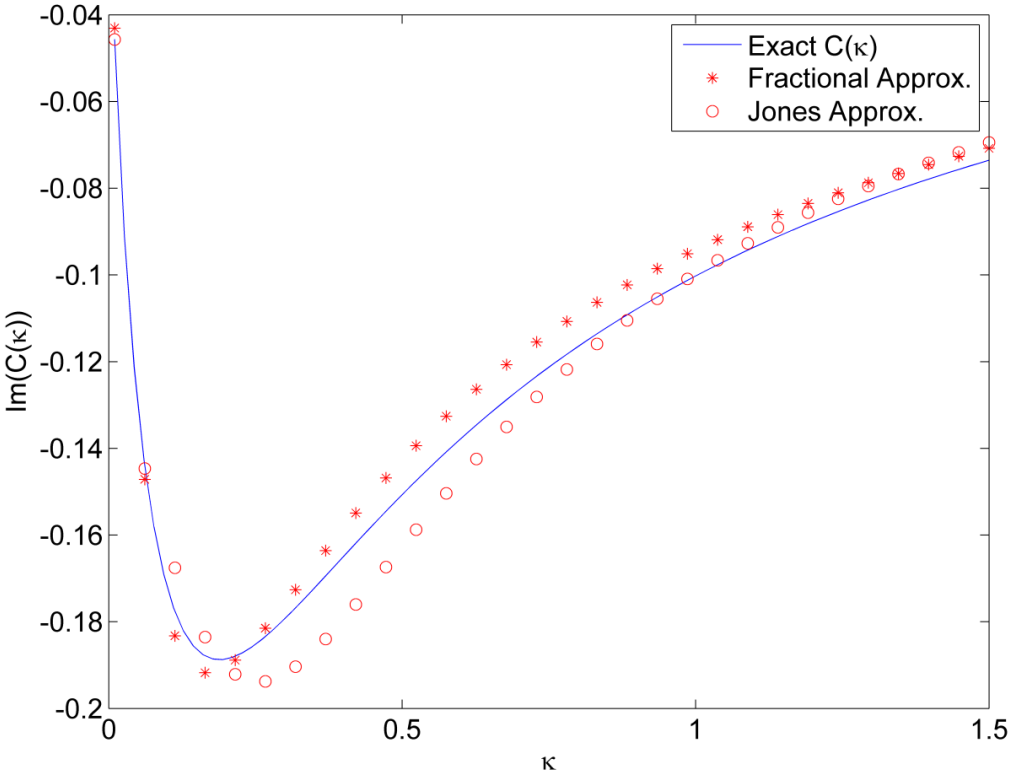


Figure 4.9: Comparison between the imaginary part of Theodorsen's function and its approximations

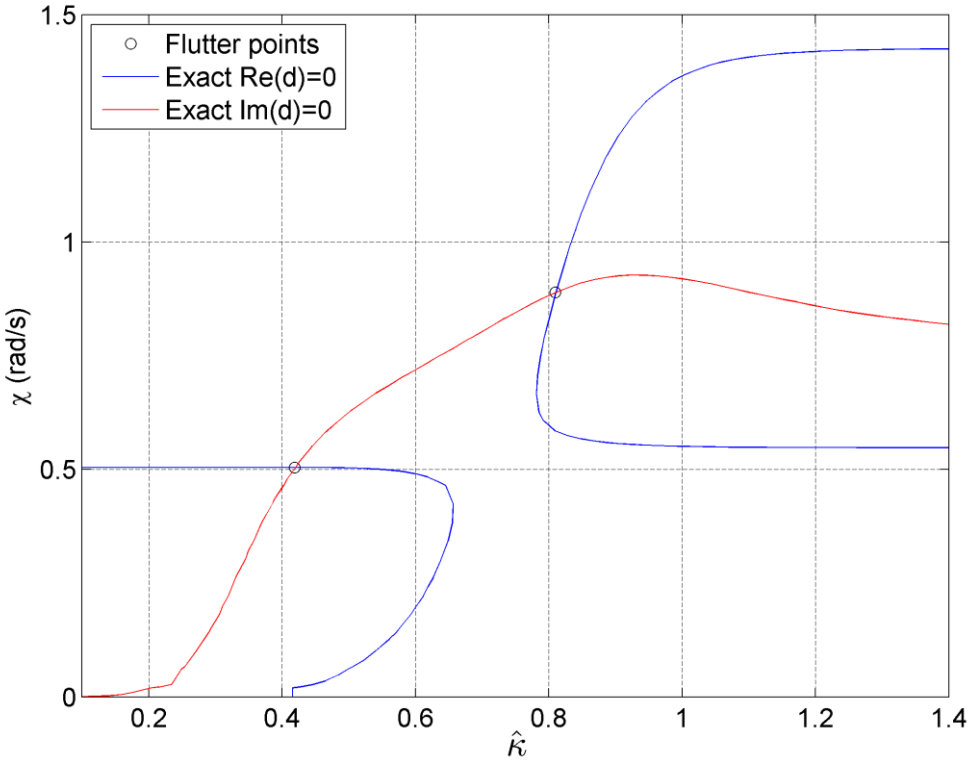


Figure 4.10: Direct flutter-point solutions for the fractional approximation superimposed over a contour plot of the exact system, in $\hat{\kappa}$.

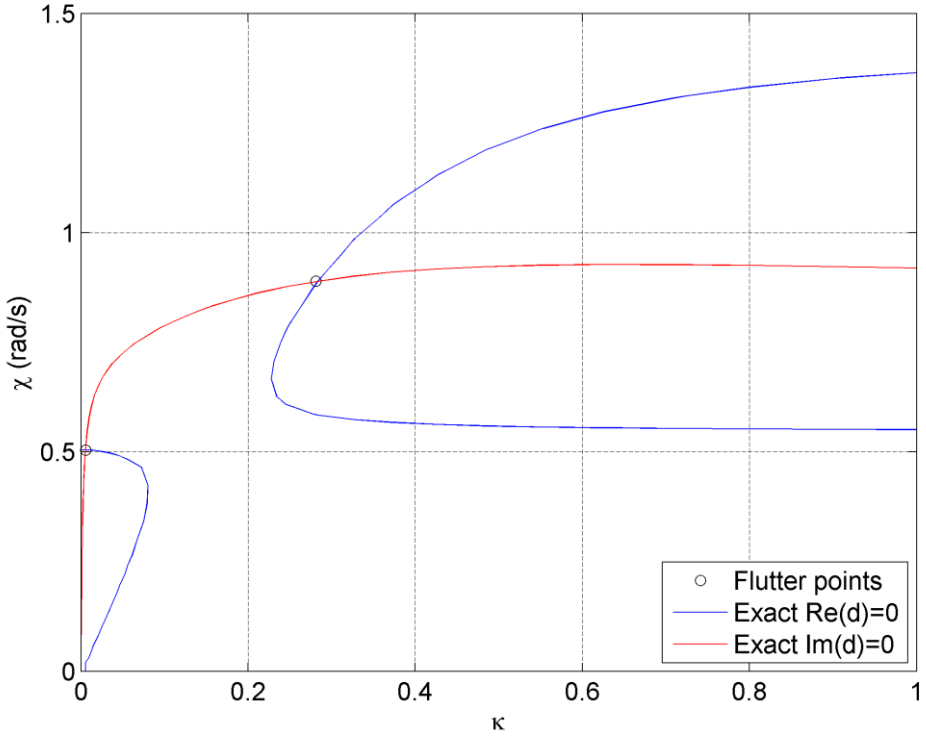


Figure 4.11: Direct flutter-point solutions for the fractional approximation superimposed over a contour plot of the exact system, in κ.

Figure 4.12 and 4.13 show a contour plot of the fractional approximation compared with a contour plot of the exact model, with the flutter points also shown. Figure 4.12 shows the solutions in terms of χ and $\hat{\kappa}$ and Figure 4.13 in terms of χ and κ. Note the difference between the approximate and exact contour plots – though the flutter points are the same. These differences arise because in the process of rearranging the fractional model into a polynomial model we had to multiply by $\kappa^2\left(1+2F(i\kappa)^{\beta}\right)$ (see Chapter 2). This factor then affects the determinant of the model and thus its contour plot. Despite this, the real contours do show some affinity to each other in areas far away from the flutter point (e.g. at high airspeeds). This is exactly what would we would expect, because away from the flutter points the real contours approximate the natural frequencies of the system (see Chapter 3) – which should be unchanged by the form of the system. This could in fact be a useful test of whether the exact contour plot accurately approximates the modal frequency paths of the system. We postulate that the difference between the contour plot of the real system and an approximation of that system gives an indication of the local accuracy of either of these contour plots in approximating the modal frequency paths of the system.

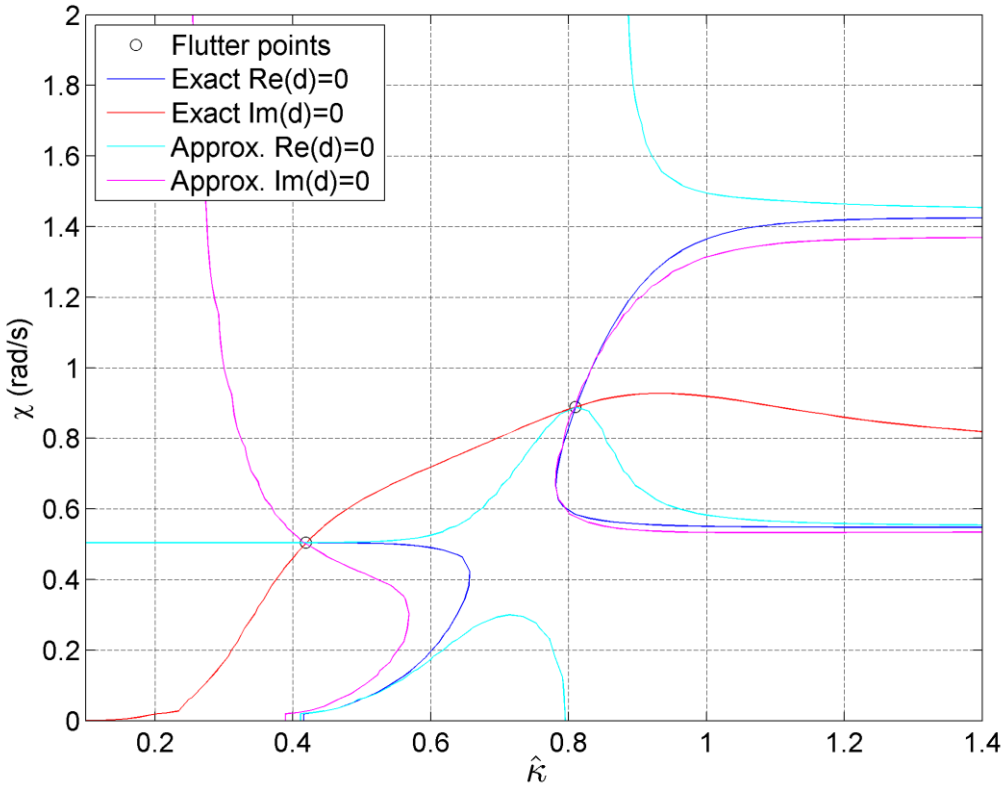


Figure 4.12: Contour plot of the fractional approximation compared with a contour plot of the exact model, in $\hat{\kappa}$.

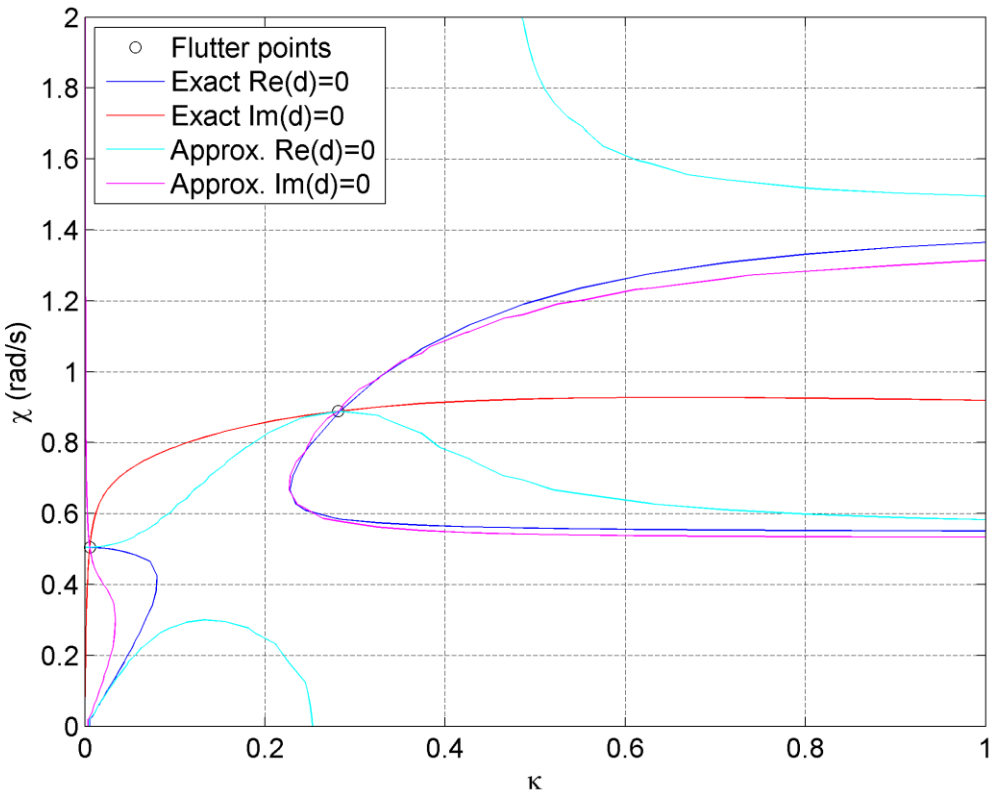


Figure 4.13: Contour plot of the fractional approximation compared with a contour plot of the exact model, in κ.

As a test of this postulate, Figure 4.14 shows the modal frequency paths of the exact Theodorsen system, superimposed on the real contours of the fractional approximation and exact system. The modal frequencies were computed by solving the τ-λ form of the exact Theodorsen system for λ for a given τ. This data was then converted into χ and κ. It can be seen that our postulate appears to be true, but is conservative. Where the real contours of the approximate system draw close to those of the exact system, the modal frequencies are indeed well-approximated by the real contours of the exact system. However, even where the approximate and exact contours are relatively far away (e.g. near $\kappa = 0.5$), the exact real contours still approximate the modal frequency paths reasonably well. Nevertheless, the small discrepancy between the modal frequencies and the exact real contours near $\kappa = 0.5$ could possibly be linked with the distance between the approximate and exact contours in this area, as though this discrepancy is small it is significantly larger than the discrepancy that it observed when the approximate and exact contours are closer together (e.g. past $\kappa = 0.8$). This is an interesting area for future investigation, particularly of an

abstract or theoretical nature. Applying this postulate into a useful algorithm would be difficult, as one has to find an adequate definition of the distance between the approximate and exact real contours – there will usually be multiple contours passing over a given κ, and for some values the contours may not exist. Nevertheless there is also potential in this area too.

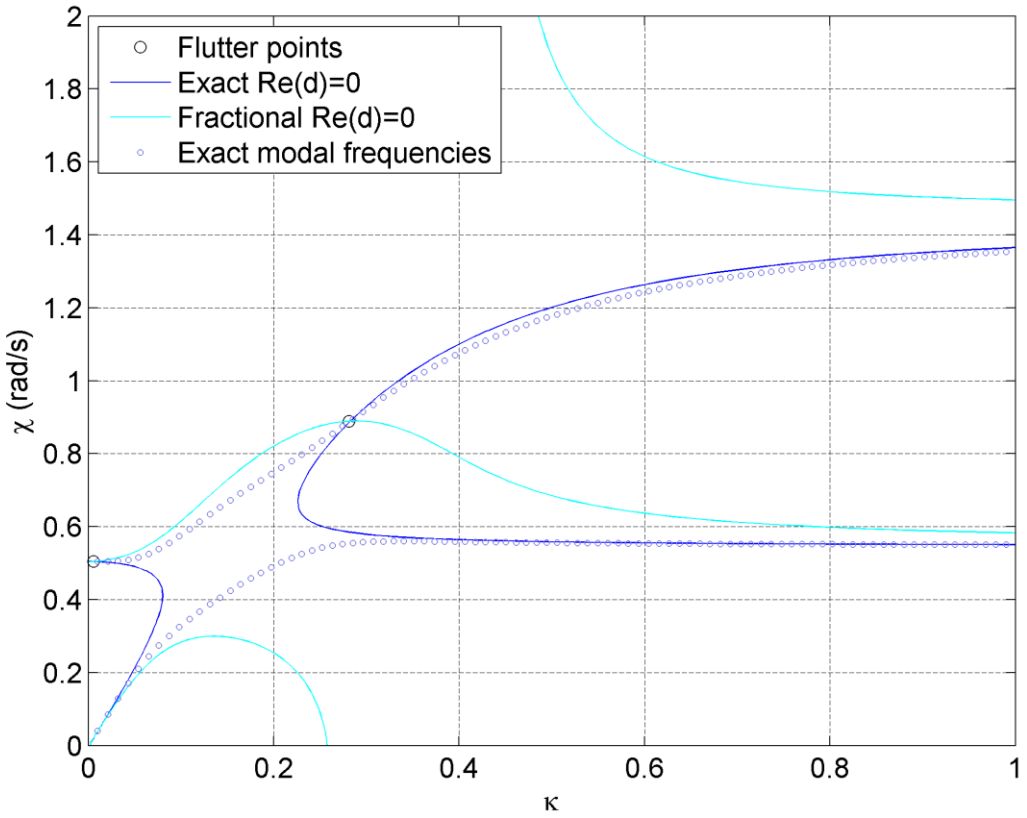


Figure 4.14: Modal frequency paths of the exact Theodorsen system, superimposed on the real contours of the fractional approximation and exact system.

As shown, the fractional approximation is very accurate. However, the computational cost of performing the operator determinant method on the linearised fractional approximation is very high. The operator determinants before compression are of size 2304×2304, and after compression they are still of size 232×232. Running MATLAB R2014b on an Intel i7-4770 with 3.40 GHz processor, 16.0 GB RAM and a 64-bit operating system, it takes an average of 20.8 seconds to complete the operator determinant method and compute the flutter points of the system. Approximately 98% of this time is taken up by the compression process – the extraction of the finite regular part of the operator determinants. Only 1% is expended on the initialisation routine (defining the system matrices and operator

determinants), and less than 1% (0.08 seconds) on the solution of the generalised eigenvalue problem in the operator determinants. This is very interesting, and suggests that efforts to increase the efficiency of the operator determinant method for polynomial problems should be focused on the development of faster compression algorithms, or else on the devising of compact nonsingular linearisations that do not require compression. The actual solution of the operator determinant GEP is very fast. As a point of interest, we may compare our algorithm to the one published by Muhič and Plestenjak [8] as this thesis was being written, which uses the same numerical method to compute the eigenvalues of the operator determinant GEP (the QZ algorithm) but implemented differently. Their algorithm takes an average of 21.2 seconds to compute the flutter points, which is not perceptibly different to our algorithm.

Consider now Jones' approximation. Figure 4.15 shows the direct flutter-point solutions for this approximation, superimposed over contour plots of the exact system. Again, the flutter points computed by the direct solver are not visually different to those that may be seen on the contour plot, and are identical to the flutter points from the contour plot to three decimal places at least. Figure 4.16 shows a contour plot of the Jones' approximation compared with a contour plot of the exact model, with the flutter points also shown. We see again that the imaginary contours in these two plots are significantly different. However, the same phenomenon occurs of the exact and approximate real contours converging in areas where the real contour / modal frequency approximation would be expected to be good. Figure 4.17 shows real contours of Jones' approximation and exact system alongside the modal frequency paths of the exact Theodorsen system. We reach the same general conclusion: the modal frequencies are well-approximated in areas where the exact and approximate real contours are in close proximity, but they are also well-approximated elsewhere. Some small changes in accuracy may be correlated with the distance between the exact and approximate contours, but further analysis is needed to ascertain the relationship in more detail.

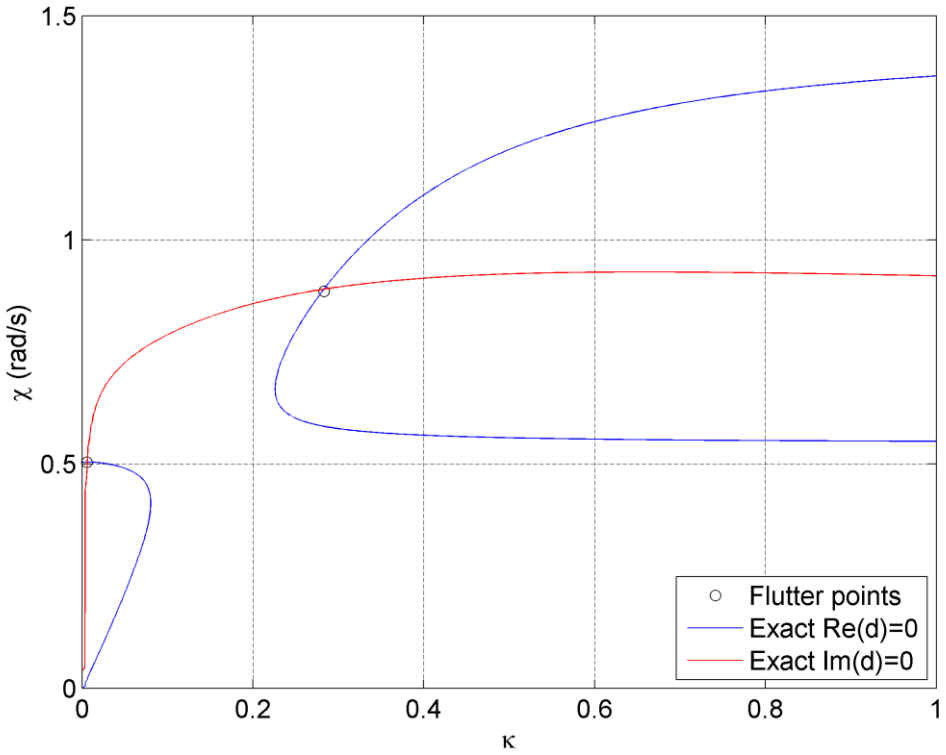


Figure 4.15: Direct flutter-point solutions for Jones' approximation superimposed over a contour plot of the exact system

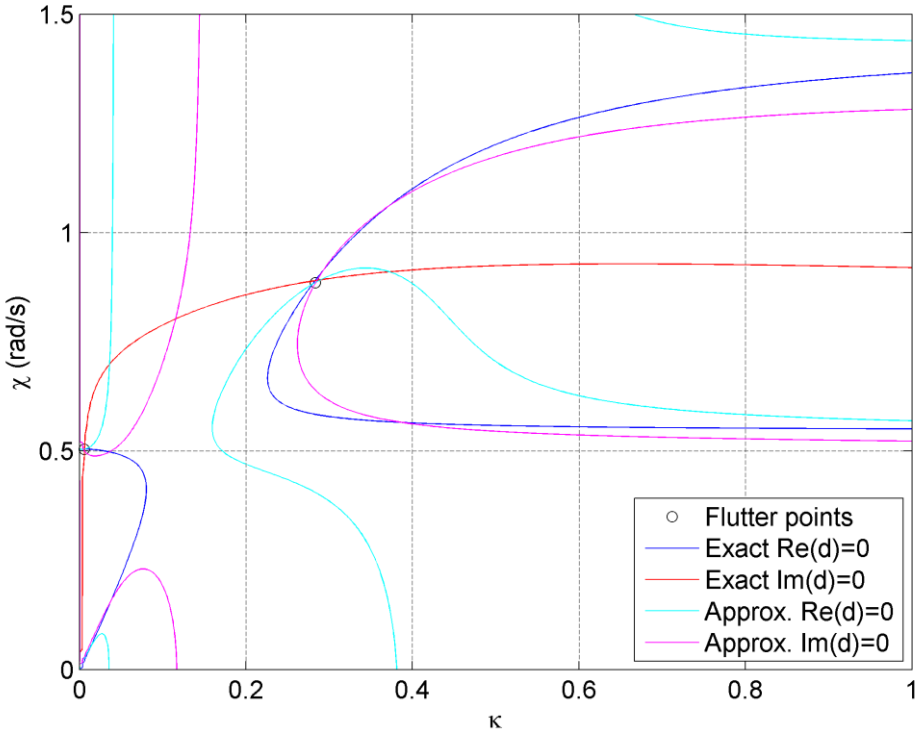


Figure 4.16: Contour plot of Jones' approximation compared with a contour plot of the exact model.

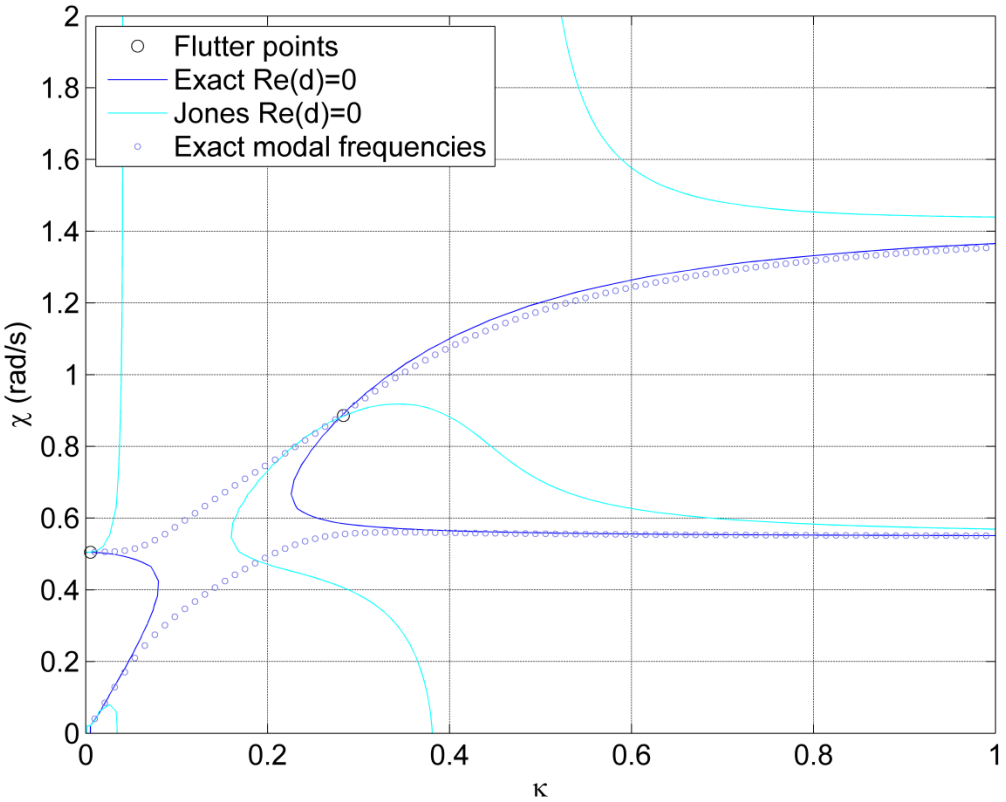


Figure 4.17: Modal frequency paths of the exact Theodorsen system, superimposed on the real contours of Jones' approximation and exact system.

As can be seen in Figure 4.15, Jones' approximation is accurate, and in an industrial application would be more than suitable. But the great advantage of Jones' approximation of the fractional one is its low computational cost. The operator determinants before compression are of size 400×400, and after compression they are of size 53×53. Running on the same Intel Pentium i7-4770, it takes an average of 0.19 seconds to complete the operator determinant method and compute the flutter points of the system. Again, the vast majority of this time (86%) is spent on extracting the finite regular part of the operator determinant pencils. Initialising the variables and defining the operator determinants takes 12%, and solving the operator determinant pencil only 2% (0.003 seconds). The algorithm to the one published by Muhič and Plestenjak [8] takes an average of 0.23 seconds to compute the flutter points: slightly longer than our method.

It is clear from these results that Jones' approximation is far more appropriate for industrial aeroelastic analysis than the fractional approximation: the difference in accuracy between

the fractional and Jones' approximation in negligible, but solving Jones' approximation is over an order of magnitude faster. The fast computation time for this approximation will allow it to be used in in optimisation or other parameter-search routines. The fact that the solver is direct is a big advantage in this respect: it is not necessary to provide initial guesses or search grids that have to be changed based on the parameter values. We anticipate that optimisation or parameter search routines will be the most significant application of this solver – for a once-off simulation one can use a contour plot, which gives more contextual information. However, other applications may yet arise.

4.5 OTHER LINEAR SOLVERS

In this chapter we concentrated on the operator determinant method as a way of solving linear problems. All our other structured solvers reduced the system to a linear one and then solved it by this method. However, other methods have been developed for the solution of these linear problems. Several of these methods actually represent an application of a general nonlinear method (e.g. Newton's method) to these linear problems, and for this reason we discuss them in Chapter 6. The use of these methods is not recommended for linear problems as they are not robust. If the system is reasonably small then the operator determinant method is convenient and reasonably fast, and if the system is large then there are several other recently-developed methods that can be applied. In terms of generally-applicable methods, there are a set of Sylvester-Arnoldi type methods [28] and a number of subspace methods (Jacobi-Davidson [11,29,30], Harmonic Rayleigh–Ritz method [31]). Note, however, that all of these methods currently have only been applied to two-parameter problems, though extensions are possible in some cases. For particular classes of two-parameter problems (weakly elliptic problems, right-definite problems), there are also a few continuation methods [12,32]. Incidentally, most of these methods in fact arise from the same research group centred at the University of Ljubljana in Slovenia. We present a brief overview of the two sets of general methods: Sylvester-Arnoldi type and subspace methods. We will also discuss these methods (and compare them to the standard operator determinant method) as part of our overview and assessment in Chapter 7.

4.5.1 Sylvester-Arnoldi

The Sylvester-Arnoldi type methods are a set of closely-related algorithms that are only valid for two-parameter problems, but are fast and can handle large systems. They still use operator determinants to reformulate the system into a set of generalised eigenvalue problems (see Section 4.2), but instead on solving this GEP at face value, the solution procedure is optimised based on the given knowledge that the operator determinants consist of a difference of two Kronecker products. Some ingenious reductions are obtained; for example, the each step of a Krylov subspace procedure reduces to the solution of a Sylvester equation. As there are several methods of solving the GEP, this approach produces several related algorithms. However, because the method is still being applied to the GEP, well-known deflation procedures can be used to find all the eigenvalues of the system. Extensions to n-parameter systems may not be possible due to the fact that the method relied on being able to reformulate the GEP into certain forms of simple and well-known matrix equations such as the Sylvester equation.

4.5.2 Subspace methods

Subspace methods for one-parameter eigenvalue problems are based around generating a series of linear spaces that eventually approximate one of the system's eigenspaces (the linear space of eigenvectors corresponding to a given eigenvalue). The Jacobi-Davidson and Rayleigh-Ritz methods are well-known one-parameter subspace methods, which can be generalised to apply to two-parameter systems [11,29–31], though this is not without significant difficulties [30]. An extension to n-parameter systems is known to be possible, but the algebra involved may become increasingly unwieldy as n increases, unless a general representation can be found. Note that these methods are applied directly to the multiparameter eigenvalue problem, and do not involve computing the operator determinants.

4.6 CONCLUDING REMARKS

In this chapter we have considered the solution of structured aeroelastic flutter problems. In terms of our methodological development, we first presented the operator determinant method, initially for a two-parameter system and then for a general n-parameter system.

We then presented two forms of linearisation (standard linearisation and quasi-linearisation) for quadratic problems and (by extension) polynomial problems. We also discussed other linear solvers that have recently been developed – we will discuss these further in Chapter 7. Note that all these solvers and linearisation methods are already known in multiparameter literature and are not a novel theoretical contribution on our part. However, we did investigate the use of perturbations to transform problems with continuous spectra to those with discrete spectra (Section 4.2.6), allowing these problems to be solved with the direct solvers that we have been describing. We also noted in Section 4.3.6 that the two-parameter compression algorithm can be applied successfully to the first three operator determinants of the n-parameter singular problem, something which has not been noticed in the literature and which opens an interesting avenue for future research.

However, the purpose of this chapter was not the development of new structured multiparameter solvers, but the application of these solvers to aeroelastic problems. Initially, we solved several simple polynomial systems (steady and quasisteady models) using these direct solvers, and obtained solutions which we then validated. The importance of these solvers should not be underestimated: apart from the use of algebraic manipulation to the compute the roots of the flutter determinant, something which is found in classical flutter analysis and applies only to trivially small problems, no flutter problem has ever been solved with a direct solver before. Previous methods have involved an iterative component, or a search along one of the eigenvalue axis (e.g. the modal damping plot). The introduction of direct solution techniques opens up an entirely new horizon for flutter research.

Having established these solvers, we then considered problems of high polynomial order – the systems arising from the approximation of Theodorsen's function with a rational or fractional-order function. These systems were solved via the same linearisation and operator determinant methods, the results were validated, and we investigated the computation times and distribution of computational effort over the solution process. We concluded that Jones' approximation to Theodorsen's function was far superior for our purposes. The solution of the problems with approximate Theodorsen aerodynamics is a significant step forward: these solvers essentially represent a form of semi-structured direct

solver, something we will discuss in Chapter 5. Again, the possibility that there might exist direct solvers (even reasonably-accurate approximate ones) for models using Theodorsen aerodynamics has not been previously considered in aeroelastic literature. Though the computation times are larger than we would like (0.2 seconds for Jones' approximation), the existence of such solvers could have significant implications for a wide variety of aeroelastic simulations – from NASTRAN's aeroelasticity solver to flutter-point optimisation routines or real-time flight envelope prediction. The direct solvers presented in Chapter 4.4 are of industrial relevance. We note that the solution of these approximate Theodorsen problems has theoretical implications as well, as they are considerately more complex than the problems that have been considered to date. They now provide a motivation for the consideration of higher-order polynomial multiparameter problems in theoretical literature, something which was not previously present.

Overall the solution methods that we have presented in this chapter represent a significant development in the study of flutter or other instability problems. The potential for multiparameter direct solvers to be used to solve flutter problems has not been recognised in prior literature. As we have shown, they are both useful in themselves, and will also be essential we consider more complex systems in the following chapters.

4.7 REFERENCES

[1] Slivnik, T., and Tomšič, G., 1986, "A numerical method for the solution of two-parameter eigenvalue problems," Journal of Computational and Applied Mathematics, **15**(1), pp. 109–115.

[2] Muhič, A., and Plestenjak, B., 2009, "On the singular two-parameter eigenvalue problem," Electronic Journal of Linear Algebra, **18**, pp. 420–437.

[3] Muhič, A., and Plestenjak, B., 2010, "On the quadratic two-parameter eigenvalue problem and its linearization," Linear Algebra and its Applications, **432**(10), pp. 2529–2542.

[4] Bernstein, D. S., 2009, Matrix mathematics theory, facts, and formulas, Princeton University Press, Princeton, New Jersey, USA.

[5] Polyanin, A. D., and Manžirov, A. V., 2007, Handbook of mathematics for engineers and scienctists, Chapman & Hall/CRC, Boca Raton, Florida, USA.

[6] Hochstenbach, M. E., Muhič, A., and Plestenjak, B., 2012, "On linearizations of the quadratic two-parameter eigenvalue problem," Linear Algebra and its Applications, **436**(8), pp. 2725–2743.

[7] Hochstenbach, M. E., and Plestenjak, B., 2003, "Backward error, condition numbers, and pseudospectra for the multiparameter eigenvalue problem," Linear Algebra and its Applications, **375**, pp. 63–81.

[8] Plestenjak, B., and Muhič, A., 2014, MultiParEig, University of Ljubljana, Ljubljana, Slovenia.

[9] Arora, S., and Barak, B., 2009, Computational complexity: a modern approach, Cambridge University Press, Cambridge, UK.

[10] Zarowski, C. J., 2004, An Introduction to Numerical Analysis for Electrical and Computer Engineers, John Wiley & Sons, New York, New York State, USA.

[11] Hochstenbach, M. E., and Plestenjak, B., 2002, "A Jacobi-Davidson Type Method for a Right Definite Two-Parameter Eigenvalue Problem," SIAM Journal on Matrix Analysis and Applications, **24**(2), pp. 392–410.

[12] Plestenjak, B., 2001, "A continuation method for a weakly elliptic two-parameter eigenvalue problem," IMA Journal of Numerical Analysis, **21**(1), pp. 199–216.

[13] Quarteroni, A., Sacco, R., and Saleri, F., 2007, Numerical mathematics, Springer-Verlag Berlin, Berlin, Germany.

[14] Higham, N. J., Mackey, D. S., Tisseur, F., and Garvey, S. D., 2008, "Scaling, sensitivity and stability in the numerical solution of quadratic eigenvalue problems," International Journal for Numerical Methods in Engineering, **73**(3), pp. 344–360.

[15] Fan, H.-Y., Lin, W.-W., and Van Dooren, P., 2004, "Normwise Scaling of Second Order Polynomial Matrices," SIAM Journal on Matrix Analysis and Applications, **26**(1), pp. 252–256.

[16] Košir, T., 1994, "Finite-dimensional multiparameter spectral theory: the nonderogatory case," Linear Algebra and its Applications, **212-213**, pp. 45–70.

[17] Binding, P., and Browne, P. J., 1989, "Two parameter eigenvalue problems for matrices," Linear Algebra and its Applications, **113**, pp. 139–157.

[18] Binding, P., and Košir, T., 1996, “Second Root Vectors for Multiparameter Eigenvalue Problems of Fredholm Type,” Transactions of the American Mathematical Society, **348**(1), pp. 229–249.

[19] Atkinson, F. V., 1968, “Multiparameter spectral theory,” Bulletin of the American Mathematical Society, **74**(1), pp. 1–28.

[20] Atkinson, F. V., 1972, Multiparameter Eigenvalue Problems: Matrices and Compact Operators, Academic Press, New York, New York State, USA.

[21] Rynne, B. P., 1988, “Multiparameter spectral theory and Taylor’s joint spectrum in Hilbert space,” Proceedings of the Edinburgh Mathematical Society, **31**(01), p. 127.

[22] Anton, H., 1977, Elementary Linear Algebra, John Wiley & Sons, New York, New York State, USA.

[23] Eiselt, H. A., and Sandblom, C.-L., 2007, Linear Programming and its Applications, Springer-Verlag Berlin, Berlin, Germany.

[24] Trefethen, L. N., and Bau, D., 1997, Numerical linear algebra, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, USA.

[25] Swinney, D., 1989, “A fractional calculus model of aeroelasticity,” Master’s Thesis, Air Force Institute of Technology.

[26] Jones, R., 1938, Operational treatment of the nonuniform-lift theory in airplane dynamics, National Advisory Committee for Aeronautics, Washington, Washington D.C., USA.

[27] Vinnikov, V., 1989, “Complete description of determinantal representations of smooth irreducible curves,” Linear Algebra and its Applications, **125**, pp. 103–140.

[28] Meerbergen, K., and Plestenjak, B., 2014, A Sylvester-Arnoldi type method for the generalized eigenvalue problem with two-by-two operator determinants, Department of Computer Science, Katholieke Universiteit Leuven, Leuven, Belgium.

[29] Meerbergen, K., Schröder, C., and Voss, H., 2013, “A Jacobi-Davidson method for two-real-parameter nonlinear eigenvalue problems arising from delay-differential equations,” Numerical Linear Algebra with Applications, **20**(5), pp. 852–868.

[30] Hochstenbach, M. E., Kosir, T., and Plestenjak, B., 2004, “A Jacobi-Davidson Type Method for the Two-Parameter Eigenvalue Problem,” SIAM Journal on Matrix Analysis and Applications, **26**(2), pp. 477–497.

[31] Hochstenbach, M. E., and Plestenjak, B., 2008, “Harmonic Rayleigh–Ritz extraction for the multiparameter eigenvalue problem,” Electronic Transactions on Numerical Analysis, **29**, pp. 81–96.

[32] Plestenjak, B., 2000, “A Continuation Method for a Right Definite Two-Parameter Eigenvalue Problem,” SIAM Journal on Matrix Analysis and Applications, **21**(4), pp. 1163–1184.

Chapter 5
Semi-structured systems

5.1 INTRODUCTION

In Chapter 4 we considered structured systems. We were able to devise direct solvers for these systems via the method of operator determinants. The question then arises of how to deal with a system to which an unstructured component has been added. In this chapter we will concentrate on the section model with Theodorsen aerodynamics. This system is could still be considered structured, because Theodorsen's function can be expressed using known functions that have analytical properties (the Hankel functions). However, these functions are transcendental, and given the fact that the system definition consists of a combination of polynomials and Theodorsen's function, it is improbable that any form of direct solver will ever be able to be developed. It is better to treat it as a black-box function, and apply one of two strategies: either to approximate the function with a convenient basis of the function space (e.g. polynomials) to high level of accuracy, and then solve the resulting system; or to approximate the function with a low level of accuracy and then solve the system iteratively, updating the approximation with data from the previous iterate and then solving the new system to produce the next iterate. These strategies are in some sense two sides of the same coin – depending on the accuracy of approximation, the required number of iterations before convergence can be increased or reduced. If the approximation is sufficiently accurate then only one iteration may be needed.

Of course, we have already applied our first strategy in the previous chapters of this thesis, when in Chapter 2 we approximated Theodorsen's function with a quadratic rational function and a fractional-power rational function. In Chapter 3 we were then able to solve these approximate systems directly. In this chapter we will consider adding an iterative component to these solvers, which will place our previous efforts more firmly in their context. We will also lay the groundwork for our more advanced iterative method development in Chapter 6, by giving on overview here of convergence criteria, iterate selection criteria, and other such necessary but peripheral matters.

5.2 PICARD ITERATION

5.2.1 Formulation

Consider again Eq. 2.3.15

$$\left((\mathrm{M}_0+\mathrm{G}_0)+\left(\mathrm{G}_1+\mathrm{G}_2C_\tau(\tau)\right)\tau+\mathrm{G}_3C_\tau(\tau)\tau^2-\mathrm{D}_0\lambda-\mathrm{K}_0\lambda^2\right)\mathbf{x}=\mathbf{0}, \tag{5.2.1}$$

and Eq. 2.3.13

$$\left((\mathrm{M}_0+\mathrm{G}_0)\chi^2+\left(\mathrm{G}_1+\mathrm{G}_2C\left(\frac{\chi}{\Upsilon}\right)\right)\Upsilon\chi+\mathrm{G}_3C\left(\frac{\chi}{\Upsilon}\right)\Upsilon^2-\mathrm{D}_0\chi-\mathrm{K}_0\right)\mathbf{x}=\mathbf{0}, \tag{5.2.2}$$

from Chapter 2. In both these equations, let us define a third eigenvalue parameter $\upsilon=C_\tau(\tau)$. Note that $\upsilon\in\mathbb{C}$. Taking the complex conjugate of Eq. 5.2.1 and Eq. 5.2.2 to form their complementary equations, we obtain the following two systems of equations:

$$\begin{gathered}\left((\mathrm{M}_0+\mathrm{G}_0)+(\mathrm{G}_1+\mathrm{G}_2\upsilon)\tau+\mathrm{G}_3\upsilon\tau^2-\mathrm{D}_0\lambda-\mathrm{K}_0\lambda^2\right)\mathbf{x}=\mathbf{0}\\ \left((\overline{\mathrm{M}}_0+\overline{\mathrm{G}}_0)+(\overline{\mathrm{G}}_1+\overline{\mathrm{G}}_2\bar{\upsilon})\tau+\overline{\mathrm{G}}_3\bar{\upsilon}\tau^2-\overline{\mathrm{D}}_0\lambda-\overline{\mathrm{K}}_0\lambda^2\right)\bar{\mathbf{x}}=\mathbf{0}\\ \upsilon=C_\tau(\tau),\end{gathered} \tag{5.2.3}$$

and

$$\begin{gathered}\left((\mathrm{M}_0+\mathrm{G}_0)\chi^2+(\mathrm{G}_1+\mathrm{G}_2\upsilon)\Upsilon\chi+\mathrm{G}_3\upsilon\Upsilon^2-\mathrm{D}_0\chi-\mathrm{K}_0\right)\mathbf{x}=\mathbf{0}\\ \left((\overline{\mathrm{M}}_0+\overline{\mathrm{G}}_0)\chi^2+(\overline{\mathrm{G}}_1+\overline{\mathrm{G}}_2\bar{\upsilon})\Upsilon\chi+\overline{\mathrm{G}}_3\bar{\upsilon}\Upsilon^2-\overline{\mathrm{D}}_0\chi-\overline{\mathrm{K}}_0\right)\bar{\mathbf{x}}=\mathbf{0}\\ \upsilon=C\left(\frac{\chi}{\Upsilon}\right).\end{gathered} \tag{5.2.4}$$

These are systems of two under-constrained multiparameter eigenvalue problems and a nonlinear scalar constraint equation. If υ is known (or guessed), then the two multiparameter eigenvalue problems can be solved by a standard singular direct solver. The eigenvalues obtained in this way (τ,λ or Υ,χ) can then be used to generate the next υ. This idea can be used as the basis for an iterative method. A general sketch of the algorithm for such a method is presented in Algorithm 5.1:

Algorithm 5.1 – sketch of iterative procedure

1	**initialise** $k = 1$, initial guesses Υ_0, χ_0, and $v_0 = C(\chi_0/\Upsilon_0)$
2	compute the set of solutions $\{\Upsilon_{k,(i)}\} \in \mathbb{R}$, $\{\mathbf{x}_{k,(i)}\} \in \mathbb{C}^n$ and $\{\chi_{k,(i)}\} \in \mathbb{R}$ to $\left((\mathrm{M}_0 + \mathrm{G}_0)\chi_k^2 + (\mathrm{G}_1 + \mathrm{G}_2 v)\Upsilon_k\chi_k + \mathrm{G}_3 v_{k-1}\Upsilon_k^2 - \mathrm{D}_0\chi_k - \mathrm{K}_0\right)\mathbf{x} = \mathbf{0}$ $\left((\overline{\mathrm{M}}_0 + \overline{\mathrm{G}}_0)\chi_k^2 + (\overline{\mathrm{G}}_1 + \overline{\mathrm{G}}_2\bar{v})\Upsilon_k\chi_k + \overline{\mathrm{G}}_3\bar{v}_{k-1}\Upsilon_k^2 - \overline{\mathrm{D}}_0\chi_k - \overline{\mathrm{K}}_0\right)\bar{\mathbf{x}} = \mathbf{0}$
3	select suitable eigenvalue-eigenvector pair $\Upsilon_k \in \{\Upsilon_{k,(i)}\}$, $\chi_k \in \{\chi_{k,(i)}\}$, $\mathbf{x}_k \in \{\mathbf{x}_{k,(i)}\}$
4	evaluate $v_k = C(\chi_k/\Upsilon_k)$
5	**if** convergence criteria satisfied
6	**then** return Υ_k, χ_k and $\mathbf{x}_k$
7	**else** $k = k + 1$, **goto** (2)

This is essentially a type of Picard iteration: we evaluate part of the system (Theodorsen's function) at the previous timestep, and then we solve for the remaining system at the current timestep. We could, of course, choose to evaluate even more of the system at the previous timestep – for example, the Υ_k^2 term – but given that this is polynomial and thus can be solved easily with direct solvers, there is really no motivation to do so. In terms of the approximation / iteration framework that we outlined in the Introduction, Section 5.1, this method is equivalent to approximating Theodorsen's function with a zeroth-order Taylor series approximate about the previous iterate. This is the lowest-order approximation that will still allow the iterations to converge.

We will first consider Algorithm 5.1 applied to Eq. 5.2.4, and then we will consider the potential advantages and disadvantages of using Eq. 5.2.3. A number of operations in Algorithm 5.1 deserve further elaboration: the solution procedure at line 2, the selection procedure at line 3, and the convergence test at line 5. Firstly, the solution of the two-parameter problem for $\{\Upsilon_{k,(i)}\}$, $\{\mathbf{x}_{k,(i)}\}$ and $\{\chi_{k,(i)}\}$. This can be carried out by the direct solvers that are described in Chapter 4. In our system, the matrix G_3 is singular and so the compression algorithm will be required even if a nonsingular linearisation is being used. The solution operation is the most computationally intensive part of Algorithm 5.1, and so it would be advantageous to try to minimise the computational effort required here. One thing to note is that we do not necessarily have to solve the system for all the coordinates of

all the eigenvalues, and their eigenvectors – we only need sufficient information to be able to select a suitable eigenvalue. We also need information for the convergence criterion (Algorithm 5.1 line 5), however, after a particular eigensolution has been selected it is trivial to compute the unknown components of this solution. We can thus break up the computation operation into a computation, selection and further computation:

Algorithm 5.2 – adaption to computation routine

2*	compute the set of solutions $\{\Upsilon_{k,(i)}\} \in \mathbb{R}$ to $\left((\mathrm{M}_0+\mathrm{G}_0)\chi_k^2+(\mathrm{G}_1+\mathrm{G}_2 v)\Upsilon_k\chi_k+\mathrm{G}_3 v_{k-1}\Upsilon_k^2-\mathrm{D}_0\chi_k-\mathrm{K}_0\right)\mathbf{x}=\mathbf{0}$ $\left((\overline{\mathrm{M}}_0+\overline{\mathrm{G}}_0)\chi_k^2+(\overline{\mathrm{G}}_1+\overline{\mathrm{G}}_2\bar{v})\Upsilon_k\chi_k+\overline{\mathrm{G}}_3\bar{v}_{k-1}\Upsilon_k^2-\overline{\mathrm{D}}_0\chi_k-\overline{\mathrm{K}}_0\right)\bar{\mathbf{x}}=\mathbf{0}$
3*	select suitable eigenvalue $\Upsilon_k \in \{\Upsilon_{k,(i)}\}$
4*	compute the set of solutions $\{\chi_{k,(i)}\} \in \mathbb{R}$ and $\{\mathbf{x}_{k,(i)}\} \in \mathbb{C}^n$ to $\left((\mathrm{M}_0+\mathrm{G}_0)\chi_k^2+(\mathrm{G}_1+\mathrm{G}_2 v)\Upsilon_k\chi_k+\mathrm{G}_3 v_{k-1}\Upsilon_k^2-\mathrm{D}_0\chi_k-\mathrm{K}_0\right)\mathbf{x}=\mathbf{0}$
5*	select suitable eigenvalue-eigenvector pair $\chi_k \in \{\chi_{k,(i)}\}$, $\mathbf{x}_k \in \{\mathbf{x}_{k,(i)}\}$

We can also devise a dual of Algorithm 5.2 which computes and selects χ_k initially instead of Υ_k; however, there are practical physical reasons for computing Υ_k first and using it as the basis for selection. We are generally interested in flutter points nearest to a particular airspeed (even if the frequency is quite different to the initial guess), rather than the flutter points that are at a similar frequency but a long way away in airspeed. The selector in Algorithm 5.2 (line 3) or its dual will have to be based on a single parameter, Υ or χ, and it is more useful to us to choose χ.

Note that a further selection is required in Algorithm 5.2, because there is the possibility that there might be multiple (real) solutions to the single eigenvalue problem. This corresponds to the existence multiple flutter points with the same Υ-value. In most physical systems, this should occur only very rarely. The only most useful criterion is to choose the χ-value which is closest to the previous iterate:

$$\chi_k = \underset{\chi_{k,(i)}}{\arg\min}\left|\chi_{k,(i)}-\chi_{k-1}\right|. \tag{5.2.5}$$

If this convergence criterion is being used then we will necessarily know χ_{k-1}. It does also require an initial guess for χ_0, but this is usually not difficult to provide. Algorithm 5.2 also requires a convergence criterion that does not require χ_k or $\mathbf{x}_k$. This is something of a restriction. The most robust selection procedure would be to choose the eigenvalue closest in distance to the previous iterate. One convenient distance metric is the Euclidean distance[1]:

$$\{\chi_{k+1}, \Upsilon_{k+1}\} = \underset{\{\chi_{k+1,(i)}, \Upsilon_{k+1,(i)}\}}{\arg\min} \left(\left(\Upsilon_{k+1,(i)} - \Upsilon_k\right)^2 + \left(\chi_{k+1,(i)} - \chi_k\right)^2 \right), \tag{5.2.6}$$

but there are many others. One particularly interesting class is a distances based entirely on Υ, for example, the selector:

$$\Upsilon_{k+1,(i)} = \underset{\Upsilon_{k+1,(i)}}{\arg\min} \left| \Upsilon_{k+1,(i)} - \Upsilon_k \right|. \tag{5.2.7}$$

This selection does not require χ_k to be known before selection, and so can be used in Algorithm 5.2. However, this comes at a cost of lower robustness, since the algorithm is not taking into account the size of its step in χ. In general, the distance-based selector can be expressed as:

$$\boldsymbol{\lambda}_k = \underset{\boldsymbol{\lambda}_{k,(i)}}{\arg\min} \left\| \boldsymbol{\lambda}_{k,(i)} - \boldsymbol{\lambda}_{k-1} \right\|, \tag{5.2.8}$$

where $\boldsymbol{\lambda} = [\chi, \Upsilon]$ is the eigenvalue vector and $\|\cdot\|$ is some scalar metric, such as a norm. In this form, Eq. 5.2.6 corresponds to $\|\boldsymbol{\lambda}\| = (\|\boldsymbol{\lambda}\|_2)^2$ and Eq. 5.2.7 to $\|\boldsymbol{\lambda}\| = |\Upsilon|$**.** It is possible to devise selectors that are not based on distance, but these suffer from the fact that the approximation to the eigenvalue problem is not usually a good approximation except close to the iterate on which the approximation was based. Nevertheless, there is one such selector that may be of use – one that always takes the first flutter point (the point with the lowest positive airspeed). This can be expressed as:

$$\Upsilon_{k+1,(i)} = \min \left\{ \Upsilon \middle| \Upsilon \in \Upsilon_{k+1,(i)}, \Upsilon > 0 \right\}. \tag{5.2.9}$$

[1] There is no purpose in performing the square-root operation here, as it does not affect the location of the minima.

Note that this selector does not require χ_k to be known before selection, and so can be used in Algorithm 5.2. Using this selector should result in an algorithm which converges more often to the first flutter point – the one which we are primarily interested in. This procedure is interesting, but it does raise some interesting questions about convergence. We are effectively allowing the algorithm to teleport its iterate to another area of the eigenvalue field. However, if the original system is far away from the first flutter point, then the teleported eigenvalue is probably no better than a wild guess. However, the purpose of this algorithm is not to allow far-off initial guesses to converge to the first flutter point, but to allow initial guesses that are close to the first flutter point to converge to it, as opposed to other flutter points which may technically be closer (according to whichever norm is used in Eq. 5.2.8). We will present numerical experiments showing of the effect of this criterion (and others) later in this chapter.

The final aspect of Algorithm 5.1 that needs elaboration is the convergence test at line 7. Two main classes of convergence criteria may be distinguished: those based on variable increment and those based on the residual. Increment-based methods are easy to implement in this algorithm: we simply define a tolerance (ϵ) on some metric of the increment of one or more eigenvalue variables. In the notation of Eq. 5.2.8 this corresponds to the criterion

$$\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_{k-1}\| < \epsilon \tag{5.2.10}$$

for convergence. It might seem immediately attractive to define the convergence criterion norm to be the same as the selector norm; however, this is not always helpful. It is true that if the Euclidean norm (squared) is being used in the selector then it is particularly sensible to use the Euclidean norm in the convergence criterion

$$(\Upsilon_k - \Upsilon_{k-1})^2 + (\chi_k - \chi_{k-1})^2 < \epsilon. \tag{5.2.11}$$

However, if the selector norm is based on only a single coordinate of $\boldsymbol{\lambda}$ (e.g. $\|\boldsymbol{\lambda}\| = |\Upsilon|$ as in Eq. 5.2.7) or the selector is not based on distance (e.g. Eq. 5.2.9) then it is much better to include a tolerance that includes the other variable (χ) as well. This prevents the solution in Υ from being falsely reported as converged when in fact χ is still changing rapidly. The use of a Euclidean distance tolerance in this case is also sensible. The only advantage of devising

a tolerance based on a single eigenvalue variable is that it would be able to be used in devising an algorithm that never computes any χ, but iterates only on Υ until convergence. However, there is not much point in devising an algorithm that never computes any χ_k. This is because χ_k is necessary for a sensible definition of the residual of the system, and a residual convergence criterion is necessary in order to prevent the system from reporting false attractors as converged solutions.

The theoretical justification for the increment tolerance is that as the system tends towards its fixed point, the eigenvalue variable increments should tend towards zero (though not necessarily monotonically). However, the reverse it not true: the increment tending toward zero does not guarantee that the system is tending towards its fixed point. Hence it is necessary to include another convergence criterion alongside the increment tolerance: this is where residual-based methods are useful. It is not immediately evident from Algorithm 5.1 how we might define a residual for this process: we appear to be dealing with both a scalar nonlinear equation $v_k = C(\chi_k/\Upsilon_k)$ and a multiparameter eigenvalue problem. However, considering Eq. 5.2.2 and its conjugate applied to a single iteration step, we have

$$\begin{aligned} \mathrm{T}(\Upsilon_k, \chi_k)\mathbf{x} &= \mathbf{0} \\ \overline{\mathrm{T}}(\Upsilon_k, \chi_k)\bar{\mathbf{x}} &= \mathbf{0}, \end{aligned} \tag{5.2.12}$$

where

$$\begin{aligned} \mathrm{T}(\Upsilon_k, \chi_k) = (\mathrm{M}_0 + \mathrm{G}_0)\chi_k^2 + \big(\mathrm{G}_1 + \mathrm{G}_2 C(\chi_k/\Upsilon_k)\big)\Upsilon_k\chi_k + \mathrm{G}_3 C(\chi_k/\Upsilon_k)\Upsilon_k^2 \\ - \mathrm{D}_0\chi_k - \mathrm{K}_0. \end{aligned} \tag{5.2.13}$$

We then seek a measure for the k-th residual of Eq. 5.2.12. This is easy to obtain: a standard method of measuring the residual vector for a one-parameter eigenvalue problem $\mathrm{T}(\lambda)\mathbf{x} = \mathbf{0}$ is:

$$\mathbf{r}_k = \mathrm{T}(\lambda_k)\mathbf{x}_k. \tag{5.2.14}$$

We can apply this residual definition directly to Eq. 5.2.12, because the residual vectors (as given by Eq. 5.2.14) of the two equations making up this multiparameter system are

complex conjugates of each other. Let $\mathbf{r}_{k,(i)}$ denote the residual of the i-th equation ($i \in \{1,2\}$) at the k-th timestep, and we have

$$\mathbf{r}_{k,(2)} = \overline{\mathrm{T}}_k \bar{\mathbf{x}}_k = \overline{\mathrm{T}_k \mathbf{x}_k} = \bar{\mathbf{r}}_{k,(1)}. \tag{5.2.15}$$

Any suitable norm of either $\mathbf{r}$ then provides a scalar measure of the residual, and a full convergence criterion is:

$$\|\mathrm{T}(\Upsilon_k, \chi_k)\mathbf{x}_k\| < \epsilon \tag{5.2.16}$$

where T is defined in Eq. 5.2.13 and $\|\cdot\|$ is some suitable norm. There is no reason not to use a standard 2-norm or 1-norm. These norms take the modulus of any complex components in $\mathrm{T}(\Upsilon_k, \chi_k)\mathbf{x}_k$.

Note that so far we have only dealt with absolute convergence criteria: the tolerance ϵ is in all cases a dimensional quantity, and an acceptable value for ϵ will depend on the system involved. However, the algorithms presented in this thesis are designed specifically for aeroelastic equations, where lengthscales and timescales are usually known a-priori; and so there is no urgent need to relativise these criteria. Moreover, the systems we are working with have already been nondimensionalised, providing some level of relativisation already.

5.2.2 Specific algorithms

Algorithms 5.3 and 5.4 are implementations of Algorithm 5.1, with specific selection criteria and convergence criteria. They are suitable for implementation into software. Algorithm 5.3 uses a selector and convergence criterion based on Euclidean distance, as well as a residual convergence criterion. The initial computation step computes the eigenvector and both eigenvalues. Algorithm 5.4 selects the first positive flutter point in the selector stage, and uses Euclidean distance and residual convergence criteria. The χ-value and eigenvector corresponding to the selected Υ is computed after the selection stage by modifying the routine as in Algorithm 5.2. Note that in both algorithms, the selection criterion has been formulated in words and not in the $\arg\min$ formulation of Eq. 5.6 and 5.7. This is for ease of readability. A direct analogue to Algorithms 5.3 and 5.4 can be obtained by replacing Υ and χ with τ and λ (as this is trivial we do not present such an algorithm, but in Section 5.2.3 we

will discuss a way of modifying the τ-λ to make it significantly more efficient but less robust).

Algorithm 5.3

1	**initialise** $k = 1$, initial guesses Υ_0, χ_0, tolerances ϵ_1 and ϵ_2, matrices M_0, G_0, G_1, G_2, D_0, K_0, matrix size n, Theodorsen's function $C(\cdot)$ and $\upsilon_0 = C(\chi_0/\Upsilon_0)$
2	compute the set of solutions $\{\Upsilon_{k,(i)}\} \in \mathbb{R}$, $\{\mathbf{x}_{k,(i)}\} \in \mathbb{C}^n$ and $\{\chi_{k,(i)}\} \in \mathbb{R}$ to $$\Big((\mathrm{M}_0 + \mathrm{G}_0)\chi_k^2 + (\mathrm{G}_1 + \mathrm{G}_2\upsilon_{k-1})\Upsilon_k\chi_k + \mathrm{G}_3\upsilon_{k-1}\Upsilon_k^2 - \mathrm{D}_0\chi_k - \mathrm{K}_0\Big)\mathbf{x} = \mathbf{0}$$ $$\Big((\overline{\mathrm{M}}_0 + \overline{\mathrm{G}}_0)\chi_k^2 + (\overline{\mathrm{G}}_1 + \overline{\mathrm{G}}_2\bar{\upsilon}_{k-1})\Upsilon_k\chi_k + \overline{\mathrm{G}}_3\bar{\upsilon}_{k-1}\Upsilon_k^2 - \overline{\mathrm{D}}_0\chi_k - \overline{\mathrm{K}}_0\Big)\bar{\mathbf{x}} = \mathbf{0}$$
3	select the eigenvalue-eigenvector pair $\Upsilon_k \in \{\Upsilon_{k,(i)}\}$, $\chi_k \in \{\chi_{k,(i)}\}$, $\mathbf{x}_k \in \{\mathbf{x}_{k,(i)}\}$ minimising $(\Upsilon_k - \Upsilon_{k-1})^2 + (\chi_k - \chi_{k-1})^2$ and (if more than one) also minimising $\lvert\chi_k - \chi_{k-1}\rvert$
4	evaluate $\upsilon_k = C(\chi_k/\Upsilon_k)$
5	**if** $(\Upsilon_k - \Upsilon_{k-1})^2 + (\chi_k - \chi_{k-1})^2 < \epsilon_1$ and $\left\lVert\Big((\mathrm{M}_0 + \mathrm{G}_0)\chi_k^2 + (\mathrm{G}_1 + \mathrm{G}_2\upsilon_k)\Upsilon_k\chi_k + \mathrm{G}_3\upsilon_k\Upsilon_k^2 - \mathrm{D}_0\chi_k - \mathrm{K}_0\Big)\mathbf{x}_k\right\rVert_2 < \epsilon_2$
6	**then** return Υ_k, χ_k and $\mathbf{x}_k$
7	**else** $k = k + 1$, **goto** (2)

Algorithm 5.4

1	**initialise** $k = 1$, initial guesses Υ_0, χ_0, tolerances ϵ_1 and ϵ_2, matrices M_0, G_0, G_1, G_2, D_0, K_0, matrix size n, Theodorsen's function $C(\cdot)$ and $\upsilon_0 = C(\chi_0/\Upsilon_0)$
2	compute the set of solutions $\{\Upsilon_{k,(i)}\} \in \mathbb{R}$ to $$\Big((\mathrm{M}_0 + \mathrm{G}_0)\chi_k^2 + (\mathrm{G}_1 + \mathrm{G}_2\upsilon_{k-1})\Upsilon_k\chi_k + \mathrm{G}_3\upsilon_{k-1}\Upsilon_k^2 - \mathrm{D}_0\chi_k - \mathrm{K}_0\Big)\mathbf{x} = \mathbf{0}$$ $$\Big((\overline{\mathrm{M}}_0 + \overline{\mathrm{G}}_0)\chi_k^2 + (\overline{\mathrm{G}}_1 + \overline{\mathrm{G}}_2\bar{\upsilon}_{k-1})\Upsilon_k\chi_k + \overline{\mathrm{G}}_3\bar{\upsilon}_{k-1}\Upsilon_k^2 - \overline{\mathrm{D}}_0\chi_k - \overline{\mathrm{K}}_0\Big)\bar{\mathbf{x}} = \mathbf{0}$$
3	select $\Upsilon_k \in \{\Upsilon_{k,(i)}\}$ such that $\Upsilon_k > 0$ and Υ_k is minimised
4	compute the set of solutions $\{\chi_{k,(i)}\} \in \mathbb{R}$ and $\{\mathbf{x}_{k,(i)}\} \in \mathbb{C}^n$ to $$\Big((\mathrm{M}_0 + \mathrm{G}_0)\chi_k^2 + (\mathrm{G}_1 + \mathrm{G}_2\upsilon)\Upsilon_k\chi_k + \mathrm{G}_3\upsilon_{k-1}\Upsilon_k^2 - \mathrm{D}_0\chi_k - \mathrm{K}_0\Big)\mathbf{x} = \mathbf{0}$$
5	select eigenvalue-eigenvector pair $\chi_k \in \{\chi_{k,(i)}\}$, $\mathbf{x}_k \in \{\mathbf{x}_{k,(i)}\}$ minimising $\lvert\chi_k - \chi_{k-1}\rvert$
6	evaluate $\upsilon_k = C(\chi_k/\Upsilon_k)$
7	**if** $(\Upsilon_k - \Upsilon_{k-1})^2 + (\chi_k - \chi_{k-1})^2 < \epsilon_1$ and $\left\lVert\Big((\mathrm{M}_0 + \mathrm{G}_0)\chi_k^2 + (\mathrm{G}_1 + \mathrm{G}_2\upsilon_k)\Upsilon_k\chi_k + \mathrm{G}_3\upsilon_k\Upsilon_k^2 - \mathrm{D}_0\chi_k - \mathrm{K}_0\Big)\mathbf{x}_k\right\rVert_2 < \epsilon_2$
8	**then** return Υ_k, χ_k and $\mathbf{x}_k$
9	**else** $k = k + 1$, **goto** (2)

5.2.3 A single-parameter iterative method

We have so far considered Eq. 5.2.4. Returning to Eq. 5.2.3, we can see that the structure of the scalar constraint equation $\upsilon = C^*(\tau_k)$ enables us to devise an algorithm which never calculates the second eigenvalue (in this case, λ). Algorithm 5.5 is one such algorithm, which uses a selector based on the distance in τ (cf. Eq. 5.2.7) and a convergence criterion based solely on distance in τ. However, it is questionable whether never computing the second eigenvalue parameter is worthwhile. In Algorithm 5.4, we have already devised an algorithm that does not need to evaluate χ during the main multiparameter solution, but does this afterwards for only the selected Υ_k. This secondary evaluation of χ costs only a single quadratic eigenvalue problem solution: the cost of this operation is likely to be small compared to the cost of solving the multiparameter problem at line 2. Algorithm 5.5 does eliminate this procedure, but at the expense of significantly less robust selection and convergence criteria, and more importantly, an inability to select the first flutter point. Hence a good initial guess is tantamount. Algorithm 5.5 is thus unlikely to be fit for general-purpose use, but may possibly be of use in refining an accurate initial estimate (e.g. from some form of model reduction) for a large system.

Algorithm 5.5

1	**initialise** $k = 1$, initial guesses τ_0, increment $\Delta\tau$, tolerance ϵ_1, matrices M_0, G_0, G_1, G_2, D_0, K_0, matrix size n, Theodorsen's function $C_\tau(\tau)$ and $\upsilon_0 = C_\tau(\tau_0)$
2	compute the set of solutions $\{\tau_{k,(i)}\} \in \mathbb{R}$ $$\left((\mathrm{M}_0 + \mathrm{G}_0) + (\mathrm{G}_1 + \mathrm{G}_2\upsilon_{k-1})\tau_k + \mathrm{G}_3\upsilon_{k-1}\tau_k^2 - \mathrm{D}_0\lambda - \mathrm{K}_0\lambda^2\right)\mathbf{x} = \mathbf{0}$$ $$\left((\overline{\mathrm{M}}_0 + \overline{\mathrm{G}}_0) + (\overline{\mathrm{G}}_1 + \overline{\mathrm{G}}_2\bar{\upsilon}_{k-1})\tau_k + \overline{\mathrm{G}}_3\bar{\upsilon}_{k-1}\tau_k^2 - \overline{\mathrm{D}}_0\lambda - \overline{\mathrm{K}}_0\lambda^2\right)\bar{\mathbf{x}} = \mathbf{0}$$
3	select $\tau_k \in \{\tau_{k,(i)}\}$ such that $\lvert\tau_k - \tau_{k-1}\rvert$ is minimised
4	evaluate $\upsilon_k = C_\tau(\tau_k)$
5	**if** $\lvert\tau_k - \tau_{k-1}\rvert < \epsilon_1$
6	**then** return τ_k
7	**else** $k = k + 1$, **goto** (2)

5.2.4 Numerical experiments

Algorithms 5.3 and 5.4 have been implemented in MATLAB. In this section we provide some results from these codes. However, before we commence the experiments, it is worth

noting one definition, that of the *order of convergence*. If a sequence $\{x_k\}$, $1 \leq k \leq \infty$, converges to some value α, then the order of convergence of this sequence is defined as p such that $\forall k$ there exists L satisfying [1]:

$$\frac{|x_{k+1} - \alpha|}{|x_k - \alpha|^p} \leq L. \tag{5.2.17}$$

This definition can be applied loosely to sequences generated by iterative numerical methods – loosely, because such sequences are always of finite length. Hence it is not possible to prove rigorously the convergence of a numerical method based on its output sequences; but for engineering purposes Eq. 5.2.17 applied to a finite sequence will be sufficient proof of convergence order. To determine the order of convergence, p, from a finite sequence $\{x_k\}$, we take the natural logarithm of both sides of Eq. 5 2.17:

$$\ln|x_{k+1} - \alpha| - p \ln|x_k - \alpha| \leq \ln L. \tag{5.2.18}$$

This is the equation of a straight line inequality. We estimate α as the final element of $\{x_k\}$. Hence, when $\ln|x_{k+1} - \alpha|$ is plotted against $\ln|x_k - \alpha|$, p will be the minimum gradient of the resulting line.

As a numerical experiment, we now simulate the section model with Theodorsen aerodynamics, with parameter values given in Chapter 2. We will use the Υ-χ form, Eq. 5.2.2. Figure 5.1 shows a contour plot with nine different iteration paths (from nine different initial guesses) superimposed. Five of the iteration paths converge to the flutter point: this point agrees with that of the contour plot, and can be located at $\Upsilon_F = 3.149$ Hz, $\chi_F = 0.8899$ rad/s. Four iteration paths appear to converge to the divergence point, but do not satisfy the convergence criteria even within 300 iterations. The iterations that converge to the flutter point can be shown to have first-order convergence. Figure 5.2 shows a logarithmic convergence plot for an initial guess of $\chi_0 = 10$ rad/s, $\Upsilon_0 = 10$ Hz. As can be seen, the convergence rate of this iteration is relatively uniform. The gradient of this plot can be measured as 1.0; this is the order of convergence. This is what we would have expected, as Picard iterations are well-known to have first-order convergence in the general case.

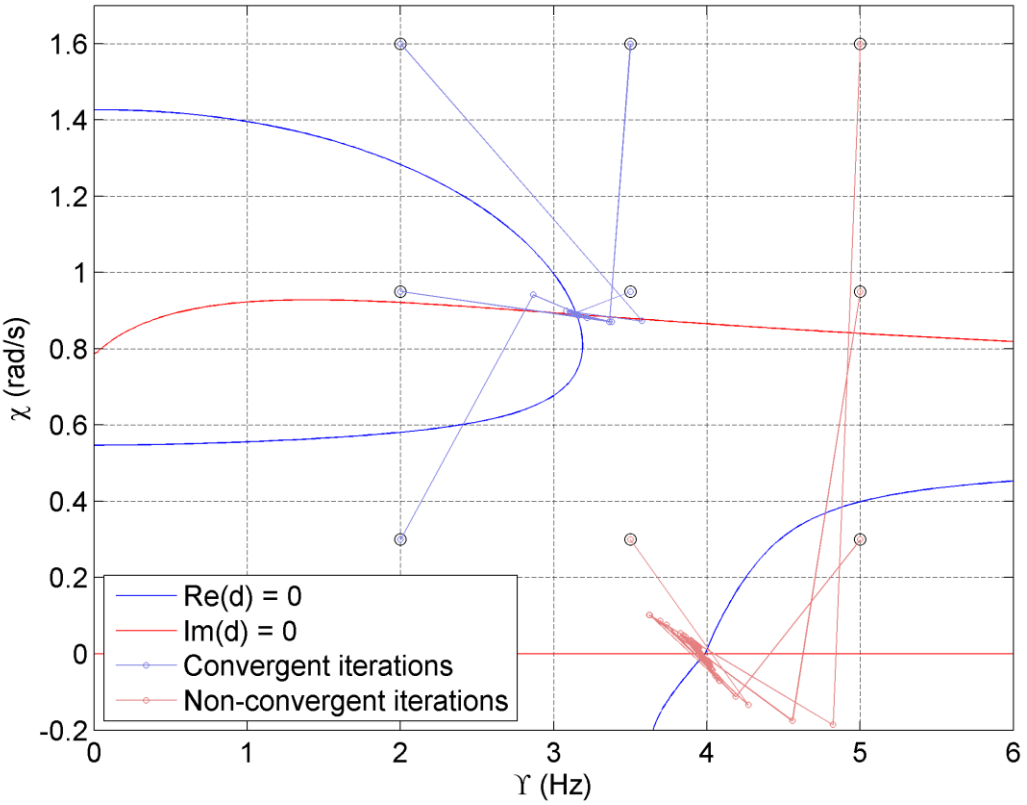


Figure 5.1: Contour plot with nine different iteration paths of the Picard iteration

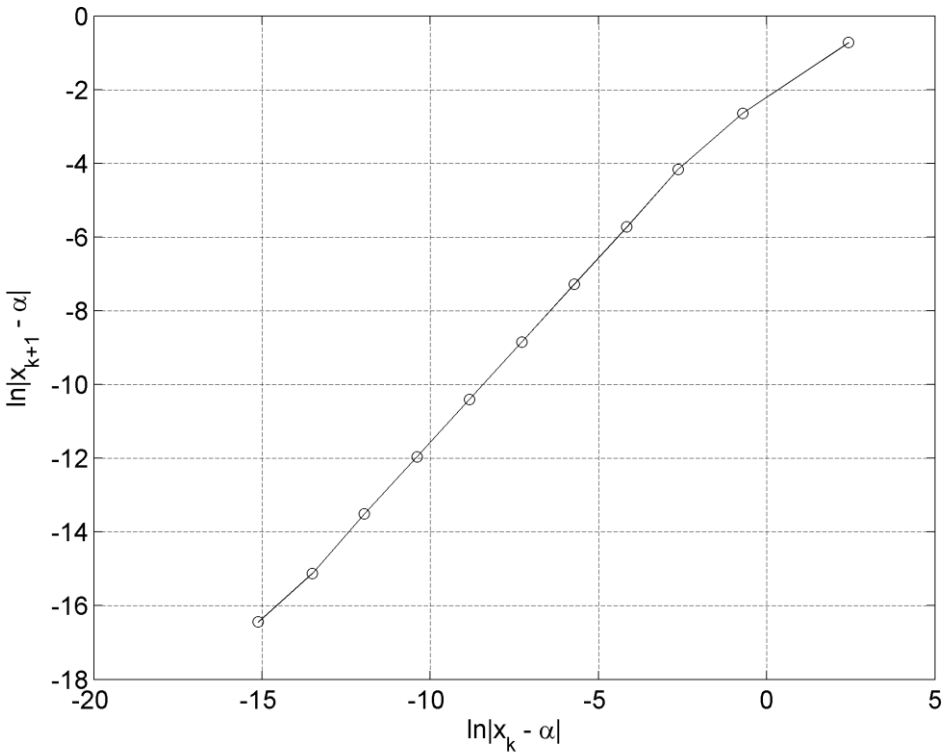


Figure 5.2: Logarithmic convergence plot for a Picard iteration converging to the flutter point ($\chi_0 = 0.5$ rad/s, $\Upsilon_0 = 5$ Hz).

The iterations to the divergence point do not converge. However, they do not diverge: they approach close to the divergence point but get trapped in an oscillation backwards and forwards over it. Figure 5.3 shows a close-up of this oscillatory behaviour, which can also be seen in the associated logarithmic convergence plot (Figure 5.4). This behaviour is almost certainly due to the fact that Theodorsen's function is nondifferentiable, and very sharply so, at the divergence point – see Chapter 3. The Picard method essentially involved taking a zeroth-order Taylor expansion of Theodorsen's function around the previous timestep (i.e. $C(\chi_k/\Upsilon_k) = C(\chi_{k-1}/\Upsilon_{k-1})$, and the nondifferentiability of Theodorsen's function at the divergence point invalidates this approximation.

To better understand the global convergence properties of this algorithm, we map out a large mesh of initial values, and test numerically whether these initial guesses result in a convergent iterative sequence (and if so, where the iteration converges to). Figure 5.5 shows the results of this process (which we will henceforth term a *numerical convergence analysis*). In general we would be disappointed with the results. There is only a thin band which converges to the first flutter point: to the left all the iterations converge to other nonphysical flutter points, and to the right all the iterations are attracted to the divergence point but do not converge.

The divergence point is clearly causing trouble in this algorithm. Fortunately, there is a way to eliminate it: if we use the eigenvalue coordinates τ and λ, then the divergence point will occur at infinite τ, and so (hopefully) will have less of an effect on the converge behaviour near the first flutter point. It is trivial to reformulate Algorithm 5.4 in terms of τ and λ, and if we do so then we obtain convergence results that are dramatically more positive. Figure 5.6 shows a numerical convergence analysis of the Picard method applied to the τ-λ form. There are no points for which the method does not converge. Figure 5.7 shows a numerical convergence analysis with a wider field of view. It is probable that the basin of attraction to the first flutter point covers almost the entirety of the upper right quarter plane. This bodes very well for the application of this method to other problems.

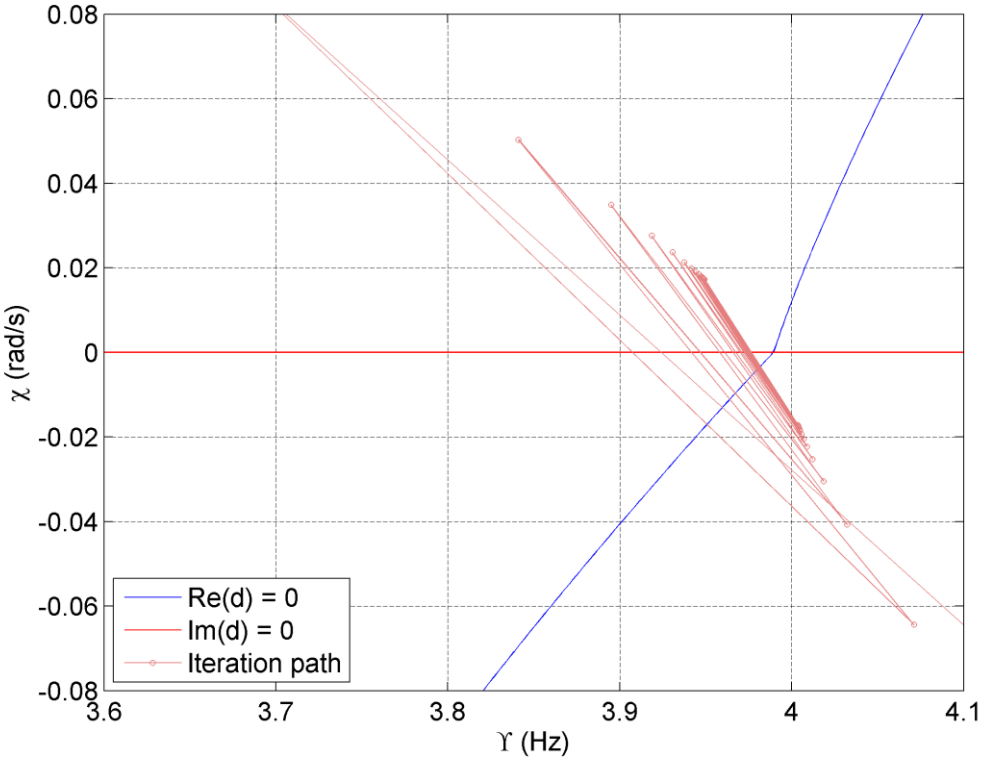


Figure 5.3: Contour plot with a Picard iteration near the divergence point, showing oscillatory behaviour

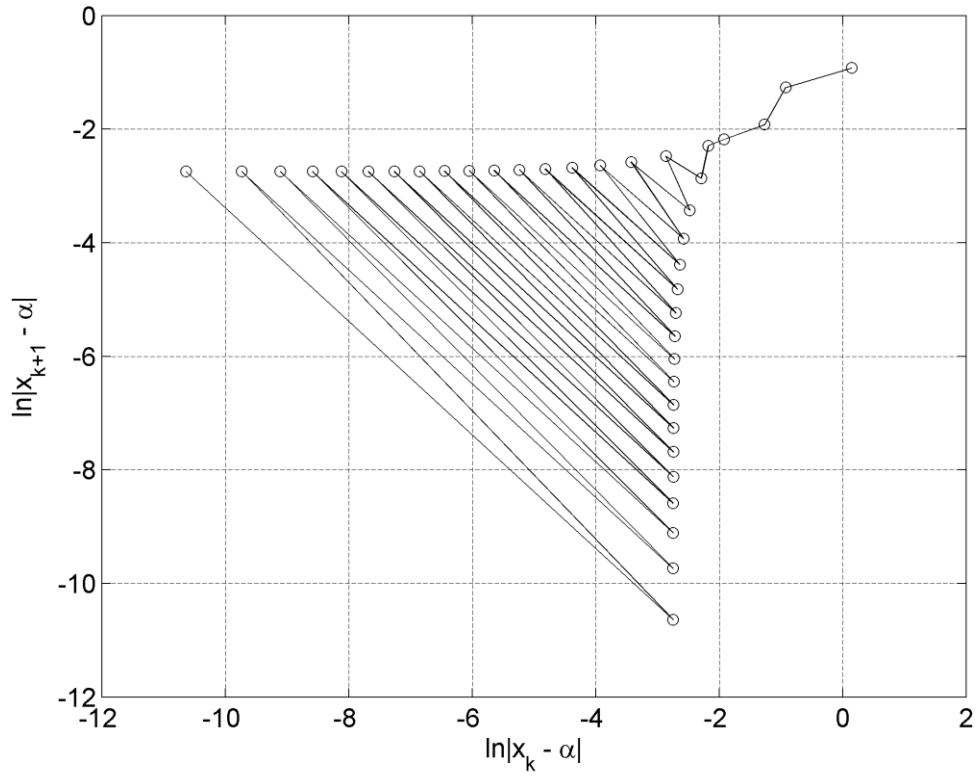


Figure 5.4: Logarithmic convergence plot for a Picard iteration converging to the divergence point

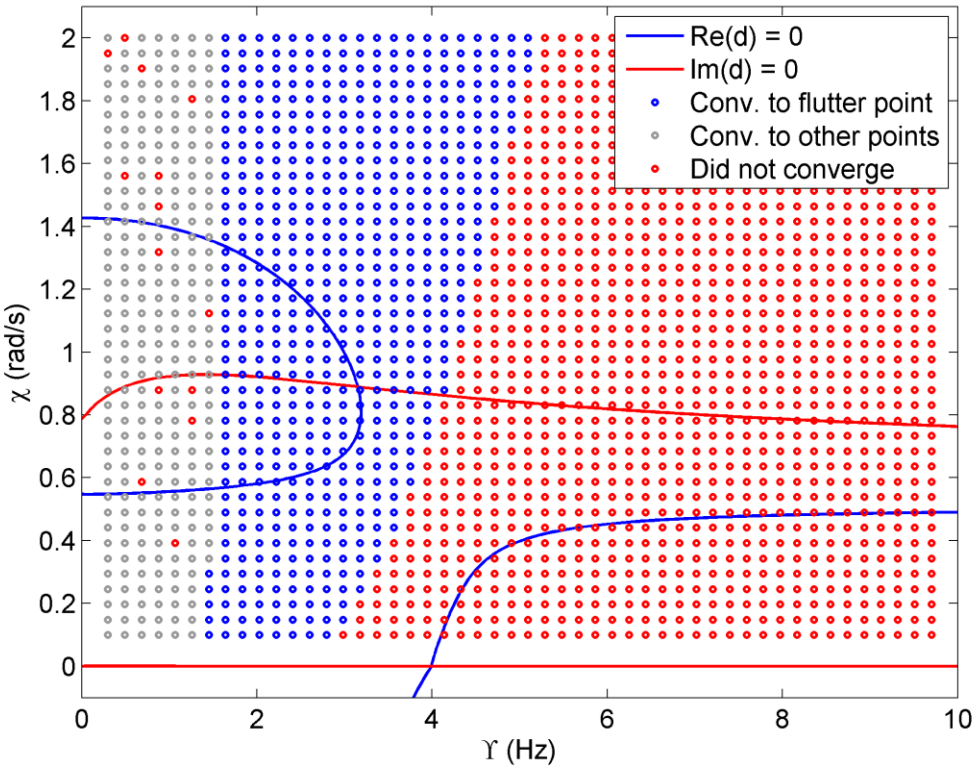


Figure 5.5: Numerical converge analysis of the Picard method (Υ-χ section model)

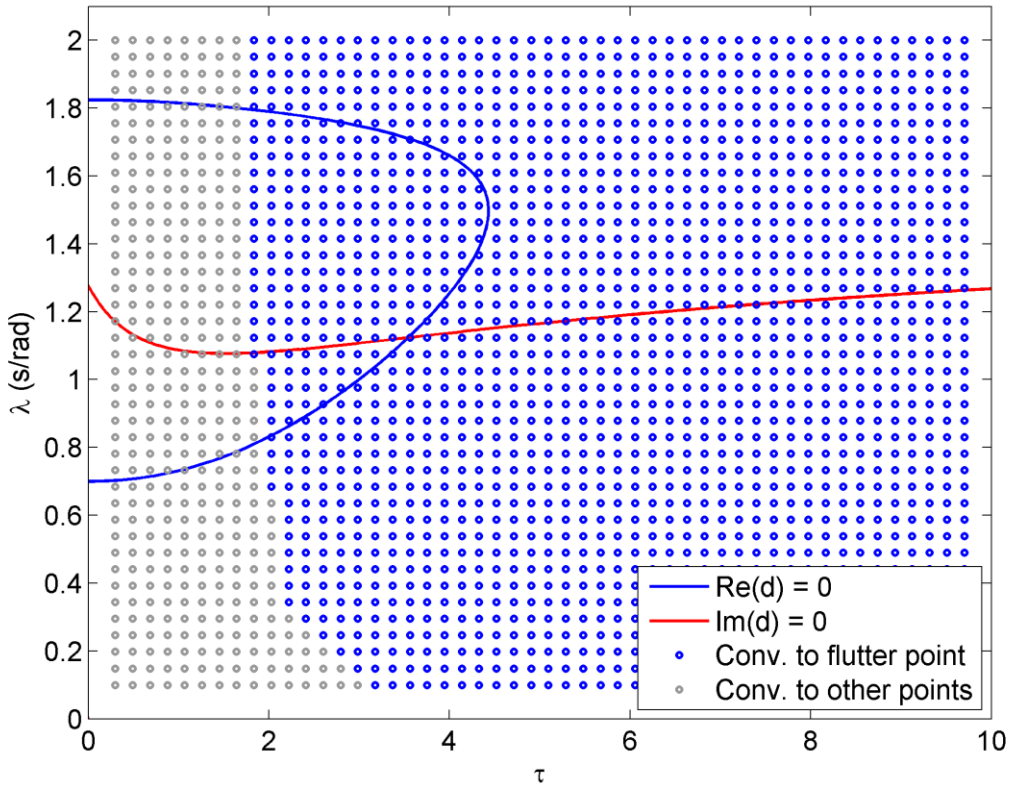


Figure 5.6: Numerical converge analysis of the Picard method (τ-λ section model).

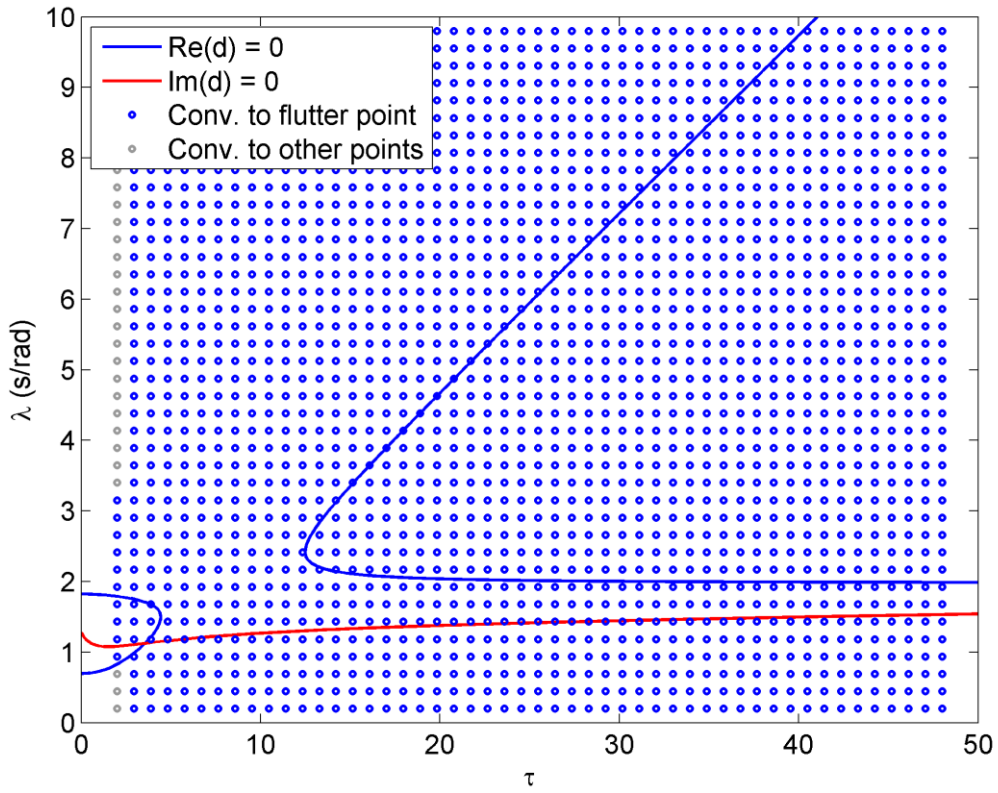


Figure 5.7: Numerical converge analysis of the Picard method (τ-λ section model, wide field of view).

5.3 NEWTON ITERATION

5.3.1 Formulation

In Section 5.2 our approximation of Theodorsen's function corresponded to a zeroth-order Taylor approximation about the previous iterate. We might logically try to use a first-order Taylor approximation instead. In this case it would be sensible to use the τ-λ form of our section model (Eq. 5.2.1), because we will then only need to expand Theodorsen's function in terms of τ (and not λ). In the Υ-χ form we would need to consider the effect of the function in both variables. So, we have

$$\Big((\mathrm{M}_0 + \mathrm{G}_0) + \big(\mathrm{G}_1 + \mathrm{G}_2 C_\tau(\tau)\big)\tau + \mathrm{G}_3 C_\tau(\tau)\tau^2 - \mathrm{D}_0\lambda - \mathrm{K}_0\lambda^2\Big)\mathbf{x} = \mathbf{0}. \tag{5.3.1}$$

Expanding $C_\tau(\tau_k)$ around τ_{k-1} we obtain:

$$C_\tau(\tau_k) = C_\tau(\tau_{k-1}) + (\tau_k - \tau_{k-1})\partial_\tau C_\tau|_{\tau_{k-1}}. \tag{5.3.2}$$

Denoting $C_{k-1} = C_\tau(\tau_{k-1})$ and $T_{k-1} = \partial_\tau C_\tau|_{\tau_{k-1}}$, we substitute this Eq. 5.3.2 into Eq. 5.3.1, with all free variables evaluated at k:

$$\Big((\mathrm{M}_0 + \mathrm{G}_0) + \big(\mathrm{G}_1 + \mathrm{G}_2(C_{k-1} + \tau_k T_{k-1} - \tau_{k-1}T_{k-1})\big)\tau_k + \mathrm{G}_3(C_{k-1} + \tau_k T_{k-1} - \tau_{k-1}T_{k-1})\tau_k^2 - \mathrm{D}_0\lambda_k - \mathrm{K}_0\lambda_k^2\Big)\mathbf{x} = \mathbf{0}. \tag{5.3.3}$$

This can be rearranged to

$$\Big((\mathrm{M}_0 + \mathrm{G}_0) + (\mathrm{G}_1 + \mathrm{G}_2 C_{k-1} - \tau_{k-1}\mathrm{G}_2 T_{k-1})\tau_k + (\mathrm{G}_3 C_{k-1} - \tau_{k-1}\mathrm{G}_3 T_{k-1} + \mathrm{G}_2 T_{k-1})\tau_k^2 + \mathrm{G}_3 T_{k-1}\tau_k^3 - \mathrm{D}_0\lambda_k - \mathrm{K}_0\lambda_k^2\Big)\mathbf{x} = \mathbf{0}. \tag{5.3.4}$$

which is a polynomial multiparameter eigenvalue problem, cubic in τ_k and quadratic in λ_k. We can linearise it with the same methods we used in Chapter 4. Defining a new eigenvector $\mathbf{q}_k = [\mathbf{x}_k;\ \tau_k\mathbf{x}_k;\ \tau_k^2\mathbf{x}_k; \lambda_k\mathbf{x}_k]$, we obtain the linear problem

$$(\mathrm{A}_k + \mathrm{B}_k\lambda_k + \mathrm{C}_k\tau_k)\mathbf{q}_k = \mathbf{0}. \tag{5.3.5}$$

with coefficients

$$\mathrm{A}_k = \begin{bmatrix} \mathrm{M}_0 + \mathrm{G}_0 & \mathrm{G}_1 + \mathrm{G}_2 C_{k-1} - \tau_{k-1}\mathrm{G}_2 T_{k-1} & 0 & -\mathrm{D}_0 \\ 0 & -I_{n\times n} & 0 & 0 \\ 0 & 0 & -I_{n\times n} & 0 \\ 0 & 0 & 0 & -I_{n\times n} \end{bmatrix}$$

$$\mathrm{B}_k = \begin{bmatrix} 0 & 0 & 0 & -\mathrm{K}_0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ I_{n\times n} & 0 & 0 & 0 \end{bmatrix} \tag{5.3.6}$$

$$\mathrm{C}_k = \begin{bmatrix} 0 & \mathrm{G}_3 C_{k-1} - \tau_{k-1}\mathrm{G}_3 T_{k-1} + \mathrm{G}_2 T_{k-1} & \mathrm{G}_3 T_{k-1} & 0 \\ I_{n\times n} & 0 & 0 & 0 \\ 0 & I_{n\times n} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

However, to evaluate the matrices A_k, B_k and C_k we required one value that we have not yet obtained: $T_{k-1} = \partial_\tau C_\tau|_{\tau_{k-1}}$. We can obtain this derivative in two ways: either by differentiating Theodorsen's function analytically, or by using finite-difference aproximations. In the general case (when we have a system with a truly unstructured component) we will have to use finite-differences, and so we use this method here. For convenience we use a first-order accurate forward difference scheme

$$\partial_\tau C_\tau = \frac{C_\tau(\tau + \Delta\tau) - C_\tau(\tau)}{\Delta\tau} \tag{5.3.7}$$

for some small increment $\Delta\tau$. Eq. 5.3.5 can be solved using the operator determinant method. This completes our formulation: we can now iterate on Eq. 5.3.5, updating A_k, B_k and C_k as the iteration progresses. Algorithm 5.6 is an implementation of the Newton iteration.

Algorithm 5.6

1	**initialise** $k = 1$, initial guesses τ_0, λ_0, increment $\Delta\tau$, tolerances ϵ_1 and ϵ_2, matrices M_0, G_0, G_1, G_2, D_0, K_0, matrix size n, Theodorsen's function $C_\tau(\tau)$
2	evaluate: $C_{k-1} = C_\tau(\tau_{k-1})$ and $T_{k-1} = 1/\Delta\tau\,(C_\tau(\tau_{k-1} + \Delta\tau) - C_{k-1})$
3	construct the matrices $$\mathrm{A}_k = \begin{bmatrix} \mathrm{M}_0 + \mathrm{G}_0 & \mathrm{G}_1 + \mathrm{G}_2 C_{k-1} - \tau_{k-1}\mathrm{G}_2 T_{k-1} & 0 & -\mathrm{D}_0 \\ 0 & -I_{n\times n} & 0 & 0 \\ 0 & 0 & -I_{n\times n} & 0 \\ 0 & 0 & 0 & -I_{n\times n} \end{bmatrix}$$ $$\mathrm{B}_k = \begin{bmatrix} 0 & 0 & 0 & -\mathrm{K}_0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ I_{n\times n} & 0 & 0 & 0 \end{bmatrix}$$ $$\mathrm{C}_k = \begin{bmatrix} 0 & \mathrm{G}_3 C_{k-1} - \tau_{k-1}\mathrm{G}_3 T_{k-1} + \mathrm{G}_2 T_{k-1} & \mathrm{G}_3 T_{k-1} & 0 \\ I_{n\times n} & 0 & 0 & 0 \\ 0 & I_{n\times n} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$
4	compute the set of solutions $\{\tau_{k,(i)}\} \in \mathbb{R}$, $\{\lambda_{k,(i)}\} \in \mathbb{R}$, $\{\mathbf{q}_{k,(i)}\} \in \mathbb{C}^{4n}$ to $$(\mathrm{A}_k + \mathrm{B}_k\lambda_k + \mathrm{C}_k\tau_k)\mathbf{q}_k = \mathbf{0}.$$ $$(\overline{\mathrm{A}}_k + \overline{\mathrm{B}}_k\lambda_k + \overline{\mathrm{C}}_k\tau_k)\overline{\mathbf{q}}_k = \mathbf{0}.$$
5	select eigenvalue-eigenvector pair $\tau_k \in \{\tau_{k,(i)}\}$, $\lambda_k \in \{\lambda_{k,(i)}\} \in \mathbb{R}$, $\mathbf{q}_k \in \{\mathbf{q}_{k,(i)}\}$ such that $d_k = (\tau_k - \tau_{k-1})^2 + (\lambda_k - \lambda_{k-1})^2$

6	evaluate $x_k = \mathbf{q}_k(1{:}n)$
7	**if** $(\tau_k - \tau_{k-1})^2 + (\lambda_k - \lambda_{k-1})^2 < \epsilon_1$ and $\left\| \left((\mathrm{M}_0 + \mathrm{G}_0) + \left(\mathrm{G}_1 + \mathrm{G}_2 C_\tau(\tau_k) \right) \tau_k + \mathrm{G}_3 C_\tau(\tau_k) \tau_k^2 - \mathrm{D}_0 \lambda_k - \mathrm{K}_0 \lambda_k^2 \right) \mathbf{x}_k \right\|_2 < \epsilon_2$
8	**then** return τ_k, λ_k and $\mathbf{x}_k$
9	**else** $k = k + 1$, **goto** (2)

5.3.2 Numerical experiments

As in Section 5.2, we now simulate the section model with Theodorsen aerodynamics, with parameter values given in Chapter 2. However, this time we are using the τ-λ form, Eq. 5.2.1, because this is the form the method is derived for. Figure 5.8 shows a contour plot with nine different iteration paths (from nine different initial guesses) superimposed. All of the iterations converge to the first flutter point at $\tau = 3.55$, $\lambda = 1.12\,\mathrm{s/rad}$. Figure 5.9 shows a logarithmic convergence plot for an iteration converging to the first flutter point, with an initial guess of $\tau_0 = 10$, $\lambda_0 = 10\,\mathrm{s/rad}$. As can be seen, the convergence rate of this iteration is relatively uniform, and very fast – only four iterations[2] are required to solve the problem to a residual and increment tolerance of 10^{-12}. The convergence is second-order: the gradient of the plot can be measured as 1.99. This is expected, as Newton-type methods are well-known to have second-order convergence [1]. The fact that we are using finite-differences to compute the derivatives has not noticeably affected the convergence rate: the finite-difference increments are sufficiently small that the computed derivative is effectively exact.

Figures 5.10 and 5.11 show a numerical convergence analysis of the semi-structured Newton's method for the section model with Theodorsen aerodynamics. The observed convergence behaviour is extremely good, and is equivalent to that of the Picard iteration. There are no points that do not convergence, and the basin of attraction around the first flutter point is unbounded upwards and to the right. Overall, we obtain the excellent convergence basins of the Picard iteration, but with second-order convergence.

[2] We can only plot three points on the plot as we must take the final iterate as the converged result for the convergence analysis (α in Eq. 5.2.17) leaving us with only four points to work with (three iterates and the initial guess), and as we are taking differences between these iterates, we only have three differences to plot.

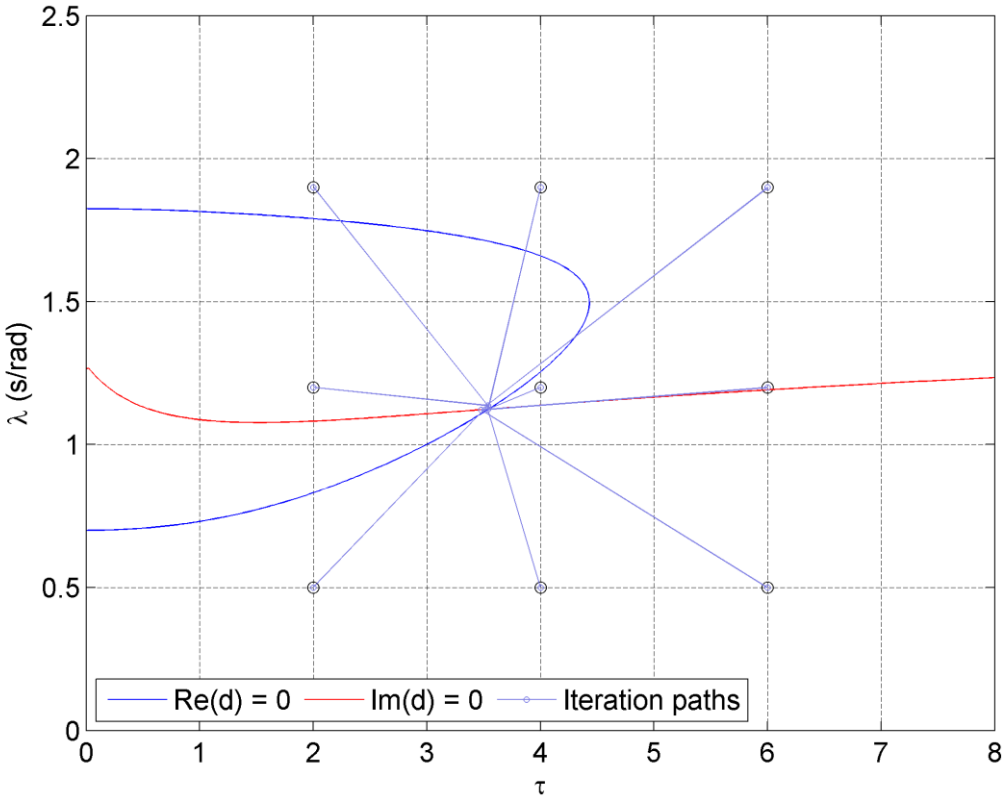


Figure 5.8: Contour plot with nine different iteration paths of the Newton's method

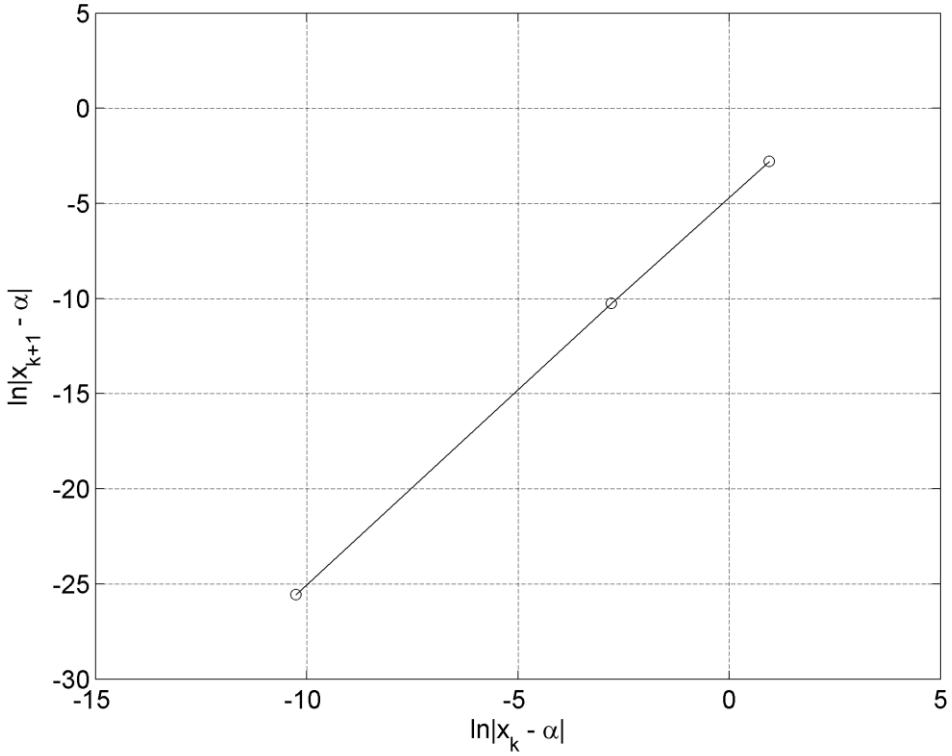


Figure 5.9: Logarithmic convergence plot for an Newton iteration converging to the first flutter point ($\chi_0 = 10$ rad/s, $\Upsilon_0 = 10$ Hz).

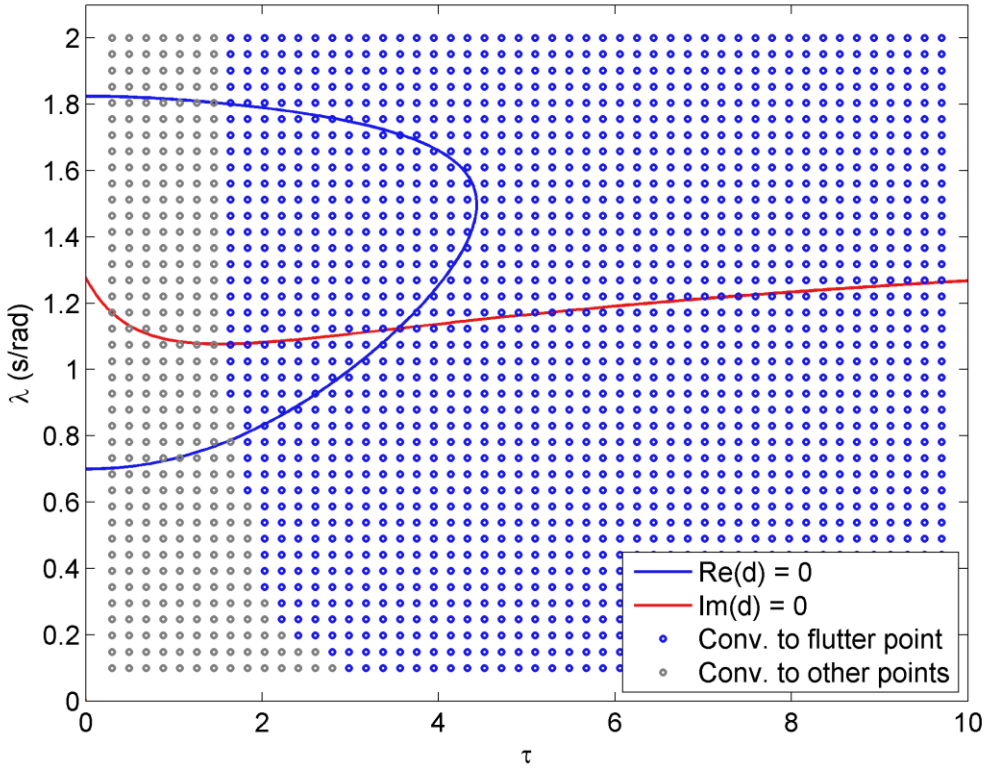


Figure 5.10: Numerical converge analysis of Newton's method (τ-λ section model).

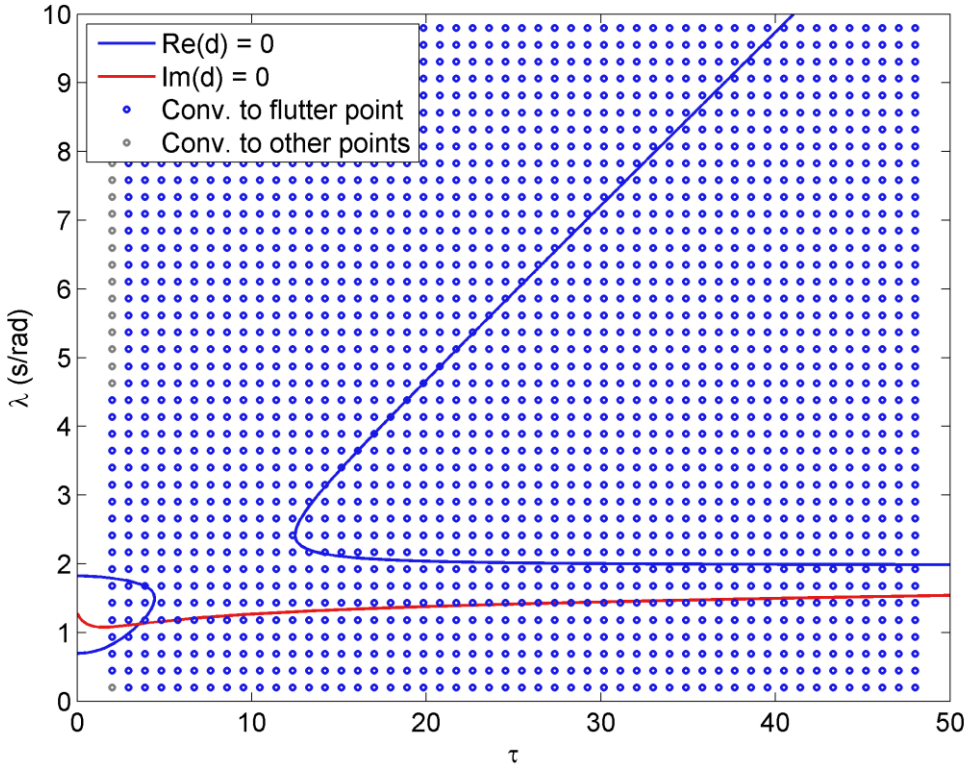


Figure 5.11: Numerical converge analysis of Newton's method (τ-λ section model, wide field of view).

5.4 HIGHER-ORDER METHODS

The previous two methods have formed a logical progression: that of increasing order in the Taylor series approximation of Theodorsen's function. We might then ask whether it is worth increasing the order of approximation any further, to second-order or beyond. For the system we have been dealing with – the section model with Theodorsen aerodynamics – there is no real motivation to do so, as the existing methods have such good convergence properties (see Section 5.2.4 and Section 5.3.2). However, for more complex systems, where there may be multiple physical flutter points, there is more reason to look at more advanced methods. We would be particularly interested in approximations that are sufficiently good that they may be able to model multiple nearby flutter points simultaneously. We have, in fact, already met such approximations, in the form of our approximations to Theodorsen's function in Chapter 2. These approximations are so good that they are able to model all physical flutter points (those inhabiting the upper right quarter plane) with a high level of accuracy – though the price is a significantly enlarged problem to solve. However, there is a subtle difference between these approximations and the Taylor approximations we have used earlier in this chapter. This is that these high-accuracy approximations are entirely global: they cannot be updated with data from a previous iteration. Hence they cannot be used in an iterative method like the previous two that we have presented.

We would ideally look for methods somewhere in between the two: that contained some globally-fixed terms, and also some local terms that could be iteratively refined. This would lead to an iterative solver that, while costly in terms of time per iteration, would converge very fast, and (more importantly) would be able to accurately identify specific flutter points to avoid or to home in on. This is the major disadvantage of the previous two solvers we have presented, in that no amount of fiddling with the selection parameters will cause their iterates to reliably converge to a given flutter point (e.g. the first flutter point). The iterates cannot reliably be made to escape the basin of attraction that they initially find themselves in. This is because the approximation of Theodorsen's function that is being used is not accurate enough to give any information about flutter points in any location outside the immediate vicinity of the current iterate. Another possible solution to this problem is to use a multi-resolution approach, with an initial global approximation to estimate the flutter point locations, and then iterative local approximations to refine these estimates. There is a

significant amount of potential here for the devising of methods with global or near-global convergence – something that would be extremely valuable in industrial applications.

However, as of yet there has been no research into developing these kinds of approximations for Theodorsen's function – the emphasis has been on devising functions that approximate Theodorsen's function globally, with no local dependence. There has previously been no motivation for considering any other forms of approximation. However, with the development of these semistructured methods for flutter problems, there is now significant impetus for the development of a much broader class of approximations.

5.5 CONLUDING REMARKS

In this chapter we have presented two iterative methods for the solution of semi-structured problems – or rather, of one particular semi-structured problem, the section model with Theodorsen aerodynamics. This is of course one of the disadvantages of semi-structured methods: by their very nature they can only be derived for systems of a given form. A completely general method would be by definition an unstructured method. The Picard iteration that we have derived applies to systems of the form

$$\big(\mathrm{A}\chi^2 + \big(\mathrm{B} + \mathrm{C}f(\chi,p)\big)\chi p + \mathrm{D}f(\chi,p)p^2 + \mathrm{E}\chi + \mathrm{F}\big)\mathbf{x} = \mathbf{0}, \qquad (5.5.1)$$

where $\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{D},\mathrm{E},\mathrm{F}$ are matrix coefficients and $f(\chi,p)$ is a scalar function of the eigenvalues χ and p. Newton's method applies to the slightly more restrictive form

$$\big(\mathrm{A}\chi^2 + \big(\mathrm{B} + \mathrm{C}f(p)\big)p + \mathrm{D}f(p)p^2 + \mathrm{E}\chi + \mathrm{F}\big)\mathbf{x} = \mathbf{0}, \qquad (5.5.2)$$

with the scalar function $f(p)$ dependent on only one eigenvalue parameter p. Though Newton's method is more restrictive, it makes up for this with faster convergence (second-order vs. first-order). The first-flutter point convergence basins for the two methods, when applied to the τ-λ form of the system, are excellent, and occupy most of the upper-right quarter plane. However, the Picard iteration performs very poorly with the Υ-χ form, something which does not bode well for the potential Newton method application to this problem. Indeed, the very poor properties of the Picard method in the Υ-χ form is the main

reason why we did not devise the Newton method for this form of the system. The fact that the Picard iteration never converges to the divergence point (thus creating a large basin of nonconvergence) is due to the discontinuity at this point; a problem which will affect our unstructured solvers as well in Chapter 6. We will discuss a method of ameliorating this problem in Chapter 6. We will also compare and contrast the semi-structured methods presented in this chapter with the approximate direct solvers (based on approximation of Theodorsen's function by a more malleable expression) in Chapter 7.

5.6 REFERENCES

[1] Quarteroni, A., Sacco, R., and Saleri, F., 2007, Numerical mathematics, Springer-Verlag Berlin, Berlin, Germany.

Chapter 6
Unstructured systems

6.1. INTRODUCTION

In this chapter we will consider methods of solution for unstructured flutter problems – problems for which very little can be said about the matrix function A in the flutter problem eigenproblem $\mathrm{A}(\chi, p)\mathbf{x} = 0$. It should be noted that no algorithms for the solution of unstructured multiparameter eigenvalue problems have ever been presented. Several iterative algorithms have been presented for the solution of linear multiparameter eigenvalue problems [1–4], as we discussed in Chapter 4, and some of these are relatively simple to generalise to the nonlinear and/or unstructured case. Furthermore, a significant number of other algorithms have been presented for the solution of the one-parameter nonlinear eigenvalue problems [5] and these can sometimes be extended to apply to multiparameter problems. We will explore both these avenues of development in this chapter.

But before we do this, it is worth asking what characteristics we desire in an algorithm for the solution of unstructured problems. We have seen in Chapter 3 that the use of the contour plot allows us to visualise the instability behaviour of even unstructured systems, and in many cases this may be the most convenient way to locate the system's flutter points. But the contour plot has two interconnected disadvantages: firstly, it is difficult to get a high level of accuracy, as the mesh must be refined to the order of accuracy required. Secondly, even reasonably low levels of accuracy are computationally expensive to obtain. An adaptive mesh refinement strategy may go some way in mitigating this expense, but there is no denying that iterative solution procedures can provide very high levels of accuracy at much lower expensive. And thirdly – and most potently – the contour plot ceases to become a useful tool (for visualisation or computation) when the eigenvalue space has dimension higher than two. Visualisation is difficult at dimension three (where the contours are now surfaces in $\mathbb{R}^3$), and effectively impossible at dimensions higher than three, and the computational cost required to mesh these higher-dimension spaces

balloons. This reason alone is sufficient to motivate the development of unstructured solvers. While in this work we are primarily interested in computing flutter points defined in $\mathbb{R}^2$, we have introduced higher-dimension flutter points in Chapter 1. We expect that the computation of these flutter points will be of interest both to researchers in aeroelasticity and to industry.

So, given the context in which we expect to use unstructured solvers, we return to our question: what characteristics would we desire of such solvers? If the system is being applied to a two-parameter flutter problem, where the use of a contour plot is feasible, then the speed of the algorithm – by which we mean a low computational cost per iteration and a high order of convergence – is important. In the higher-order case what we would really be interested in is an algorithm with global or near-global convergence properties. Global convergence is less of an issue in the two-parameter case, where we would expect to have a greater physical knowledge of the system, and probably a reasonable estimate of the flutter point locations from a coarse contour plot. However, in a higher-dimensional system it is less likely that such estimates will be available, and thus the need for global convergence strategies increases. We would also be interested in locating multiple flutter points: perhaps all of the flutter points in a certain domain, or at least, flutter points with some key characteristics (e.g. the first flutter point).

We will not, of course, present a perfect unstructured algorithm in this thesis. The algorithms that we present are, after all, the first of their kind. They should rather be seen as a starting point for future development and ultimately software. This is not a thesis in mathematics, and we will not present proofs of convergence or any other properties. Our objective in this chapter is primarily to introduce the idea that multiparameter algorithms can be developed for the solution of unstructured flutter problems – something that appears to have gone largely unrecognised in both mathematical and aeroelastic literature – and secondly, to provide a groundwork of such algorithms for future practitioners to apply and adapt to their own problems.

6.2. THE METHOD OF SUCCESSIVE LINEAR PROBLEMS (SLP)

6.2.1 Formulation

Consider the general formulation of the aeroelastic stability problem, Eq. 1.2.2:

$$\begin{aligned} \mathrm{A}(\chi,p)\mathbf{x} &= 0 \\ \overline{\mathrm{A}}(\chi,p)\bar{\mathbf{x}} &= 0. \end{aligned} \tag{6.2.1}$$

If $\mathrm{A}(\chi,p)$ and its first derivatives are continuously dependent on χ and p then we may expand $\mathrm{A}(\chi,p)$ in a first-order multivariate Taylor series about a point $[\chi_0,p_0]$ [6]:

$$\mathrm{A}(\chi,p) = \mathrm{A}(\chi_0,p_0) + (\chi-\chi_0)\partial_\chi \mathrm{A}(\chi_0,p_0) + (p-p_0)\partial_p \mathrm{A}(\chi_0,p_0). \tag{6.2.2}$$

This approximation is accurate to the first order in both variables. We can expand $\overline{\mathrm{A}}$ similarly, and noting that $\partial_x \bar{f} = \overline{\partial_x f}$, we have:

$$\overline{\mathrm{A}}(\chi,p) = \overline{\mathrm{A}}(\chi_0,p_0) + (\chi-\chi_0)\overline{\partial_\chi \mathrm{A}}(\chi_0,p_0) + (p-p_0)\overline{\partial_p \mathrm{A}}(\chi_0,p_0). \tag{6.2.3}$$

That is, if we have sufficient information to evaluate Eq. 6.2.2 then we necessarily have sufficient information to evaluate Eq. 6.2.3. This is a saving in computational cost, as only two derivatives need be evaluated. Substituting Eq. 6.2.2 and 6.2.3 into Eq. 6.2.1, we obtain

$$\begin{aligned} \Big(\mathrm{A}(\chi_0,p_0) - \chi_0\partial_\chi \mathrm{A}(\chi_0,p_0) - p_0\partial_p \mathrm{A}(\chi_0,p_0) + \chi\partial_\chi \mathrm{A}(\chi_0,p_0) \\ + p\partial_p \mathrm{A}(\chi_0,p_0)\Big)\mathbf{x} = 0 \\ \Big(\overline{\mathrm{A}}(\chi_0,p_0) - \chi_0\overline{\partial_\chi \mathrm{A}}(\chi_0,p_0) - p_0\overline{\partial_p \mathrm{A}}(\chi_0,p_0) + \chi\overline{\partial_\chi \mathrm{A}}(\chi_0,p_0) \\ + p\overline{\partial_p \mathrm{A}}(\chi_0,p_0)\Big)\bar{\mathbf{x}} = 0, \end{aligned} \tag{6.2.4}$$

which is now a linear multiparameter eigenvalue problem in χ and p, which can be solved by the Operator Determinant method (Chapter 4). For the systems we are interested in, the eigenvalue problem is generally nonsingular, and so we do not generally need the compression algorithm. Thus we devise a fixed-point iteration, Eq. 6.2.5. This iteration is essentially a generalisation of the successive linear problems algorithm for one-parameter problems as proposed by Ruhe [5]. The one-parameter case the algorithm is known to have quadratic convergence [5], and there is no reason why this should not extend to the multiparameter. Ruhe also provides the interesting observation that, when the system is

overdamped (in a way that he defines), it is possible to devise a global convergence strategy. However, it is not clear whether this strategy will apply to the multiparameter case (i.e. higher dimensional eigenvalue space).

$$\begin{aligned}\Big(\mathrm{A}(\chi_{k-1},p_{k-1})-\chi_{k-1}\partial_\chi\mathrm{A}(\chi_{k-1},p_{k-1})-p_{k-1}\partial_p\mathrm{A}(\chi_{k-1},p_{k-1})\\ +\chi_k\partial_\chi\mathrm{A}(\chi_{k-1},p_{k-1})+p_k\partial_p\mathrm{A}(\chi_{k-1},p_{k-1})\Big)\mathbf{x}=0,\\ \Big(\overline{\mathrm{A}}(\chi_{k-1},p_{k-1})-\chi_{k-1}\overline{\partial_\chi\mathrm{A}}(\chi_{k-1},p_{k-1})-p_{k-1}\overline{\partial_p\mathrm{A}}(\chi_{k-1},p_{k-1})\\ +\chi_k\overline{\partial_\chi\mathrm{A}}(\chi_{k-1},p_{k-1})+p_k\overline{\partial_p\mathrm{A}}(\chi_{k-1},p_{k-1})\Big)\bar{\mathbf{x}}=0.\end{aligned} \tag{6.2.5}$$

Eq. 6.2.5 requires the derivatives of A with respect to χ and p. If these cannot be obtained analytically, then finite-difference approximations can be used. We consider two possible schemes: a second-order accurate central difference scheme

$$\begin{aligned}\partial_p\mathrm{A}(\chi_0,p_0)&=\frac{\mathrm{A}(\chi_0,p_0+\Delta p)-\mathrm{A}(\chi_0,p_0-\Delta p)}{2\Delta p},\\ \partial_\chi\mathrm{A}(\chi_0,p_0)&=\frac{\mathrm{A}(\chi_0+\Delta\chi,p_0)-\mathrm{A}(\chi_0-\Delta\chi,p_0)}{2\Delta\chi},\end{aligned} \tag{6.2.6}$$

or a first-order accurate forward-difference scheme

$$\begin{aligned}\partial_p\mathrm{A}(\chi_0,p_0)&=\frac{\mathrm{A}(\chi_0,p_0+\Delta p)-\mathrm{A}(\chi_0,p_0)}{\Delta p},\\ \partial_\chi\mathrm{A}(\chi_0,p_0)&=\frac{\mathrm{A}(\chi_0+\Delta\chi,p_0)-\mathrm{A}(\chi_0,p_0)}{\Delta\chi}.\end{aligned} \tag{6.2.7}$$

We choose the first-order forward-difference scheme, because it involves fewer function evaluations ($\mathrm{A}(\chi_0,p_0)$ must be evaluated anyway, and so this value can be reused). The order of accuracy of the schemes is not a major issue, as we can make the increments sufficiently small that any difference in accuracy is negligible. However, in some cases it may be possible to find one of the derivatives analytically – the aerodynamic contribution can usually be expressed as a function of one parameter, which usually leaves the dependence of the system on the other parameter analytical. Algorithm 6.1 presents a sketch of a SLP algorithm. Note that the SLP method has never before been used even in abstract

multiparameter spectral theory: the development of this method has relevance both to aeroelasticity and to other fields.

Algorithm 6.1 – sketch of successive linear problems algorithm

1	**initialise** $k = 0$, initial guesses χ_0, p_0
2	evaluate $\mathrm{A}_k = \mathrm{A}(\chi_k, p_k)$, $\mathrm{P}_k = \partial_p \mathrm{A}(\chi_k, p_k)$, $\mathrm{X}_k = \partial_\chi \mathrm{A}(\chi_k, p_k)$
3	compute the set of solutions $\{\chi_{k+1,(i)}\}$, $\{\mathbf{x}_{k+1,(i)}\}$ and $\{p_{k+1,(i)}\}$ to $(\mathrm{A}_k - \mathrm{X}_k\chi_k - \mathrm{P}_k p_k + \chi_{k+1}\mathrm{X}_k + p_{k+1}\mathrm{P}_k)\mathbf{x} = 0$ $(\overline{\mathrm{A}}_k - \overline{\mathrm{X}}_k\chi_k - \overline{\mathrm{P}}_k p_k + \chi_{k+1}\overline{\mathrm{X}}_k + p_{k+1}\overline{\mathrm{P}}_k)\bar{\mathbf{x}} = 0$
4	select suitable eigenvalue-eigenvector pair, $p_{k+1} \in \{p_{k+1,(i)}\}$, $\chi_k \in \{\chi_{k+1,(i)}\}$, $\mathbf{x}_{k+1} \in \{\mathbf{x}_{k+1,(i)}\}$
5	**if** convergence criteria satisfied
6	**then** return p_k, χ_k and $\mathbf{x}_k$
7	**else** $k = k + 1$, **goto** (2)

We have already discussed the evaluation of the derivatives (line 3). However, as with the semistructured algorithms (Chapter 5), there are number of other areas that need further detail: the selection process (line 4) and the convergence criterion (line 5). However, as the concerns are exactly the same we will not dwell on them for any length. The most robust selection algorithm is one based on minimum Euclidean distance:

$$\{\chi_{k+1}, p_{k+1}\} = \underset{\{\chi_{k+1,(i)}, p_{k+1,(i)}\}}{\arg\min} \left(\left(p_{k+1,(i)} - p_k\right)^2 + \left(\chi_{k+1,(i)} - \chi_k\right)^2 \right) \tag{6.2.8}$$

For convergence criteria, it is necessary to tolerance both the residual and the increment of the eigenvalues in order to get an accurate picture of whether the iteration is converging or not. A sensible choice for the increment measure is the Euclidean distance, in which case the convergence criterion is

$$(p_k - p_{k-1})^2 + (\chi_k - \chi_{k-1})^2 < \epsilon_1 \tag{6.2.9}$$

for convergence. The residual of the system can be measured by $\|\mathrm{A}(\chi_k, p_k)\mathbf{x}_k\|$, in which case the convergence criterion is

$$\|\mathrm{A}(\chi_k, p_k)\mathbf{x}_k\| < \epsilon_2. \tag{6.2.10}$$

The reader is referred to Chapter 5 for a fuller discussion of the merits of these criteria.

6.2.2 Specific algorithms

Algorithm 6.2 is an implementation of Algorithm 6.1, with specific selection criteria and convergence criteria. Both derivatives (Algorithm 6.1 line 2) are calculated using a first-order forward difference (Eq. 6.2.7). The Euclidean distance selector is used (Eq. 6.2.8), as well as the Euclidean increment tolerance (Eq. 6.2.9) and a residual tolerance (Eq. 6.2.10). The algorithm is suitable for implementation into software. Note that in both algorithms, the selection criterion has been formulated in words and not in the $\arg\min$ formulation of Eq. 6.2.8. This is for ease of readability.

Algorithm 6.2 – specific successive linear problems algorithm

1	**initialise** $k = 0$, initial guesses χ_0, p_0, tolerances ϵ_1, ϵ_2, small increments δp, $\delta\chi$
2	evaluate $\mathrm{A}_k = \mathrm{A}(\chi_k, p_k)$,
3	evaluate $$\mathrm{P}_k = \frac{\mathrm{A}(\chi_k, p_k + \delta p) - \mathrm{A}_k}{\delta p}$$ $$\mathrm{X}_k = \frac{\mathrm{A}(\chi_k + \delta\chi, p_k) - \mathrm{A}_k}{\delta\chi}$$
4	compute the set of solutions $\{\chi_{k+1,(i)}\}$, $\{\mathbf{x}_{k+1,(i)}\}$ and $\{p_{k+1,(i)}\}$ to $$(\mathrm{A}_k - \mathrm{X}_k\chi_k - \mathrm{P}_k p_k + \chi_{k+1}\mathrm{X}_k + p_{k+1}\mathrm{P}_k)\mathbf{x} = 0$$ $$(\overline{\mathrm{A}}_k - \overline{\mathrm{X}}_k\chi_k - \overline{\mathrm{P}}_k p_k + \chi_{k+1}\overline{\mathrm{X}}_k + p_{k+1}\overline{\mathrm{P}}_k)\bar{\mathbf{x}} = 0$$
5	select eigenvalue-eigenvector pair, $p_{k+1} \in \{p_{k+1,(i)}\}$, $\chi_k \in \{\chi_{k+1,(i)}\}$, $\mathbf{x}_{k+1} \in \{\mathbf{x}_{k+1,(i)}\}$ such that $(p_{k+1,(i)} - p_k)^2 + (\chi_{k+1,(i)} - \chi_k)^2$ is minimised
6	**if** $(p_k - p_{k-1})^2 + (\chi_k - \chi_{k-1})^2 < \epsilon_1$ and $\|\mathrm{A}(\chi_k, p_k)\mathbf{x}_k\| < \epsilon_2$
7	**then** return p_k, χ_k and $\mathbf{x}_k$
8	**else** $k = k + 1$, **goto** (2)

6.2.3 Numerical experiments

We are now in a position to simulate one of the models described in Chapter 2: the section model with Theodorsen aerodynamics. We use the nondimensional form in Υ-χ with parameter values as per Chapter 2. Figure 6.1 shows a contour plot with nine different iteration paths (from nine different initial guesses) superimposed. Six of the iteration paths converge to the first flutter point, two to flutter points with negative airspeed values, and one to the divergence point. The first flutter point can be located at $\Upsilon_F = 3.149$ Hz, $\chi_F = 0.8899$ rad/s. This point agrees with the flutter point predicted by the contour plot at $\Upsilon_F = 3.15$ Hz , $\chi_F = 0.890$ rad/s. The divergence point is predicted by the algorithm to lie at $\Upsilon_D = 3.989$ Hz, which again agrees with the contour plot.

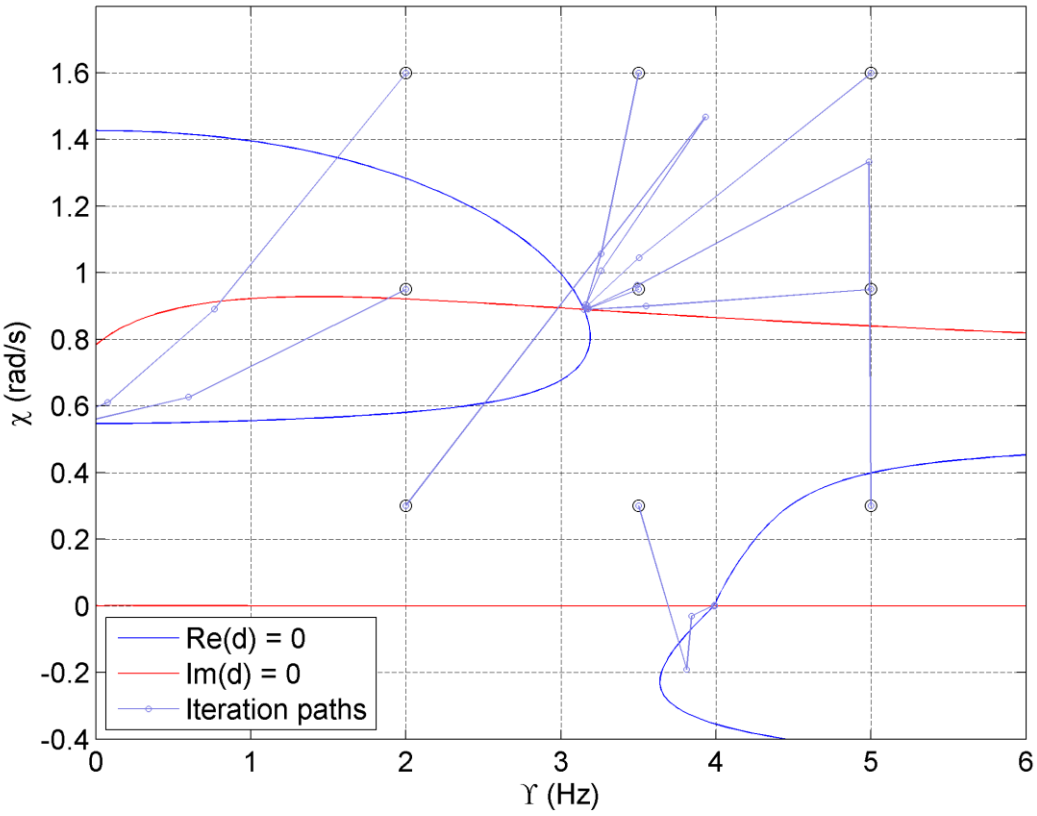


Figure 6.1: Contour plot with nine different iteration paths of the successive linear problems algorithm.

Figure 6.2 shows a logarithmic convergence plot for an initial guess of $\chi_0 = 4$ rad/s, $\Upsilon_0 = 6$ Hz. This iterative sequence converges to the first flutter point. As can be seen, the convergence rate of this iteration is relatively uniform. The gradient of this plot (order of

convergence) can be measured as 2.00. This confirms that the quadratic convergence or Ruhe's one-parameter algorithm does extend to the multiparameter case. Note that the fact that we are using finite-differences to compute the derivatives has not affected the convergence rate in any noticeable way: the finite-difference increments are sufficiently small that the computed derivative is effectively exact.

Figure 6.3 shows a logarithmic convergence plot for an initial guess of $\chi_0 = 0.3\,\mathrm{rad/s}$, $\Upsilon_0 = 10\,\mathrm{Hz}$. This iterative sequence converges to the divergence point. Note the difference in uniformity between Figure 6.2 and Figure 6.3: the convergence to the divergence point is less uniform. It also has lower order: the average gradient of Figure 6.3 is approximately 1.03. Initially it might be supposed that the divergence point represents a repeated root (to which Newton-like methods are known to have first-order convergence [7]), but a quick check reveals that this is not the case – only one singular value of $\mathrm{A}(\chi, p)$ is near zero in the vicinity of $\chi = 0\,\mathrm{rad/s}$ and $\Upsilon = 3.99\,\mathrm{Hz}$. The lower order of convergence is probably related to the fact that the first derivative of A with respect to χ is discontinuous at the divergence point, as we showed in Chapter 3. This means that the matrix function is not analytic in this vicinity, and so the Taylor expansion of Eq. 6.2.2 is not valid. However, as noted there, the use of our multiparameter algorithms to locate this divergence point is somewhat redundant, as it can be computed using one-parameter algorithms that are already well-known. However, it is possible that the problem of computing a multiparameter divergence may arise when the flutter point location is in a space of dimension higher than two (see Chapter 1). In this situation, one would have to substitute the a-priori known information about the divergence point (i.e. $\chi = 0$) into the formulation of the system. This would reduce the dimension of the system by at least 1, because χ or its equivalent would necessarily be eliminated as an eigenvariable. The reduced-order system could then be solved with the multiparameter methods, and the discontinuity in χ would be eliminated.

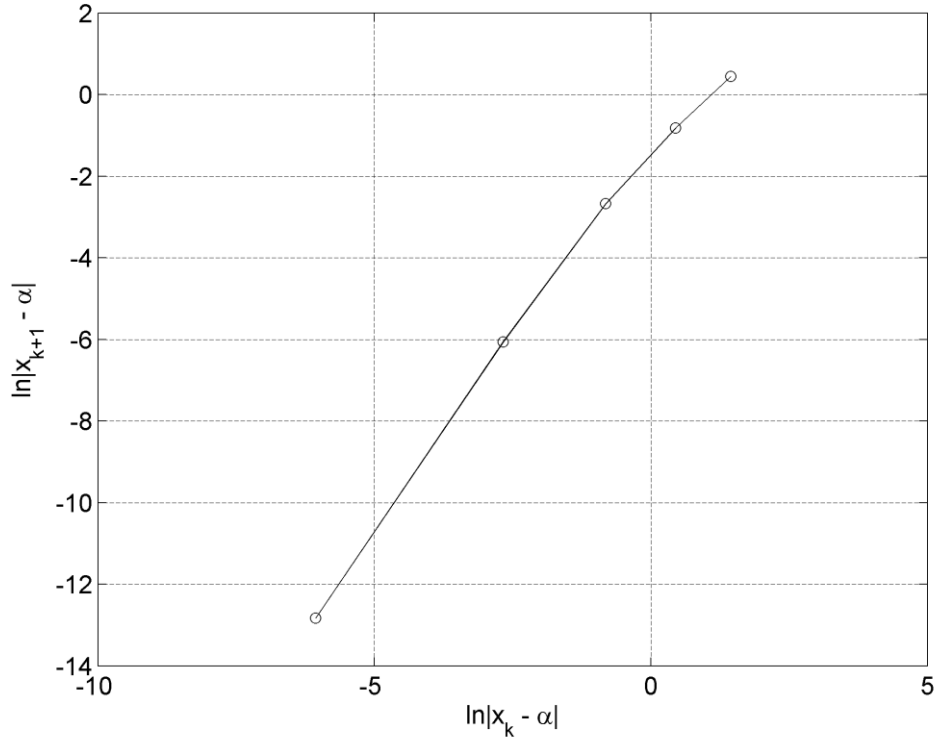


Figure 6.2: Logarithmic convergence plot for an SLP iteration converging to the flutter point ($\chi_0 = 4\ \mathrm{rad/s}, \Upsilon_0 = 6\ \mathrm{Hz}$).

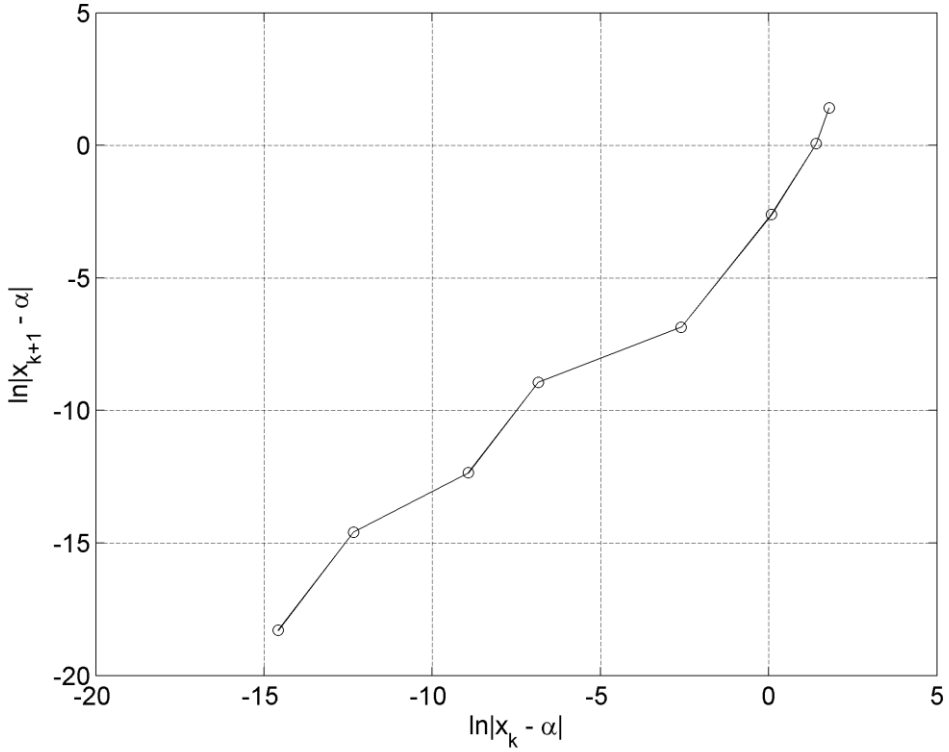


Figure 6.3: Logarithmic convergence plot for an SLP iteration converging to the flutter point ($\chi_0 = 0.3\ \mathrm{rad/s}, \Upsilon_0 = 10\ \mathrm{Hz}$).

As noted in the introduction (Section 6.1) we are interested in the convergence properties or our algorithms, particularly whether some sort of near-global convergence is available. Figures 6.4, 6.5 and 6.6 show a numerical convergence analysis of the SLP algorithm applied to the section model with Theodorsen aerodynamics. The observed convergence behaviour is very good – only a few scattered points do not convergence. There is a wide basin of attraction around the first flutter point, though it is mixed in a somewhat chaotic manner with the basin of attraction of the divergence point. As can be seen in Figure 6.5, the combined basin of attraction of the first flutter and divergence points extend far into the right-half plane, and may be unbounded. However, that there is a small basin of attraction around the restabilisation point at $\Upsilon = 93.6\ \mathrm{Hz}$. This basin can be observed in Figure 6.6, and while it is reasonably wide (with the first attracted point observed near $\Upsilon = 50\ \mathrm{Hz}$), it does not extend above $\chi = 5\ \mathrm{rad/s}$ over the surveyed Υ range. These convergence results are very encouraging, as there are no large areas of non-convergence. However, the basin of attraction of the divergence point in the upper-right quarter plane is significant, and detracts from the basin of attraction of the first flutter point (the flutter point that we would desire most to compute). In this case, we can eliminate the divergence point by using the τ-λ form instead of the Υ-χ form. Figures 6.7 and 6.8 show numerical convergence analyses for the τ-λ form. As can be seen, the results are very positive. The basin of attraction for the first flutter point probably occupies the entire upper-right quarter plane, apart from the small bands near the axes. The τ-λ form is clearly superior if it is the first flutter point that is of interest, for this system at any rate. Overall, the convergence results for the SLP algorithm are very positive – there are no significant areas of non-convergence.

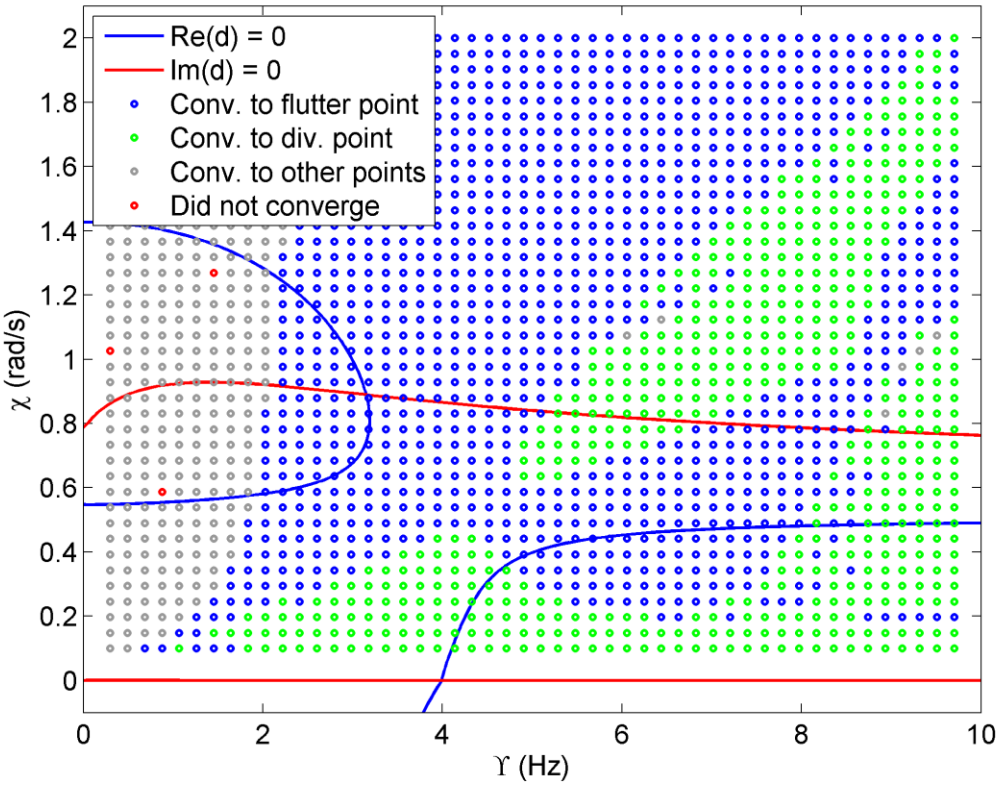


Figure 6.4: Numerical convergence analysis of the SLP algorithm (Υ-χ section model).

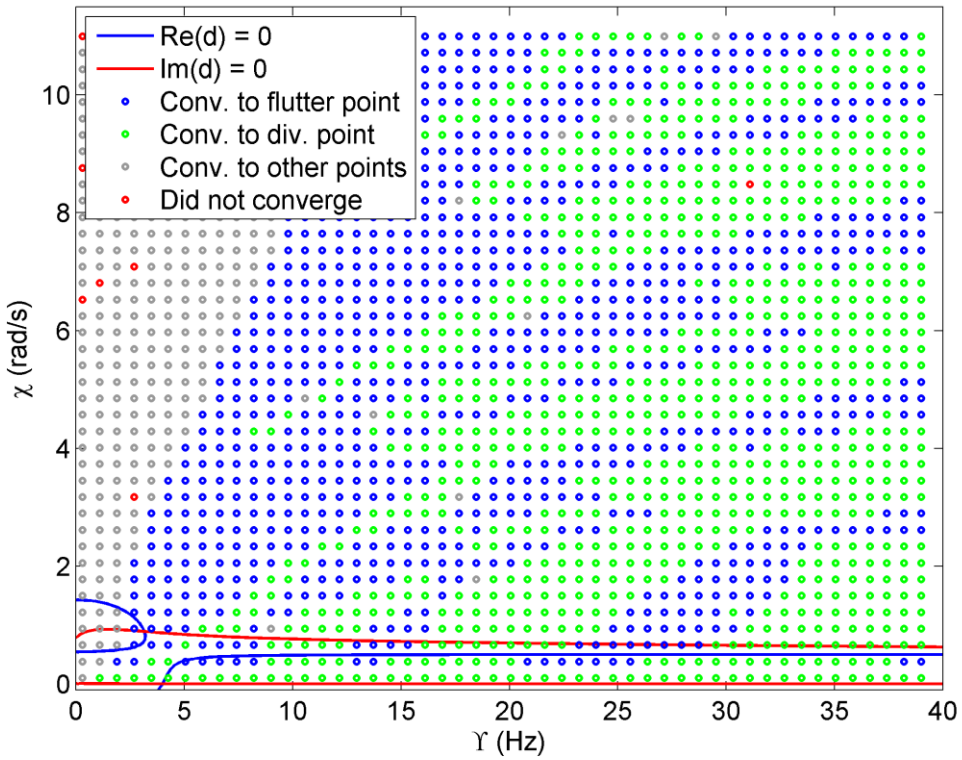


Figure 6.5: Numerical convergence analysis of the SLP algorithm (Υ-χ section model, wide field of view).

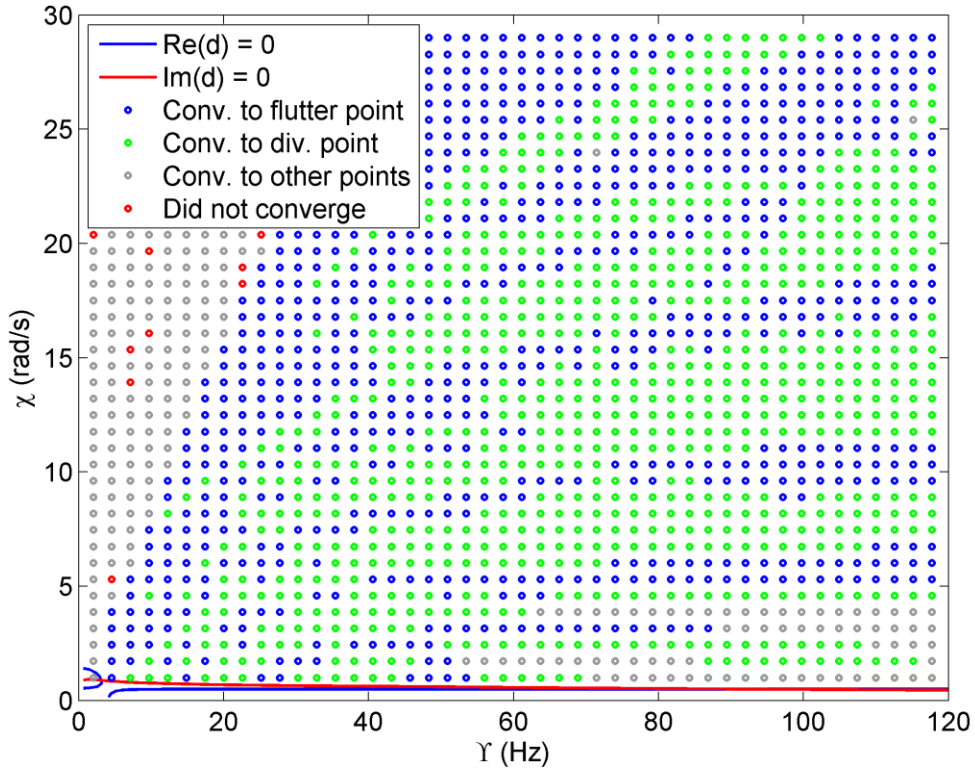


Figure 6.6: Numerical convergence analysis of the SLP algorithm (Υ-χ section model, very wide field of view).

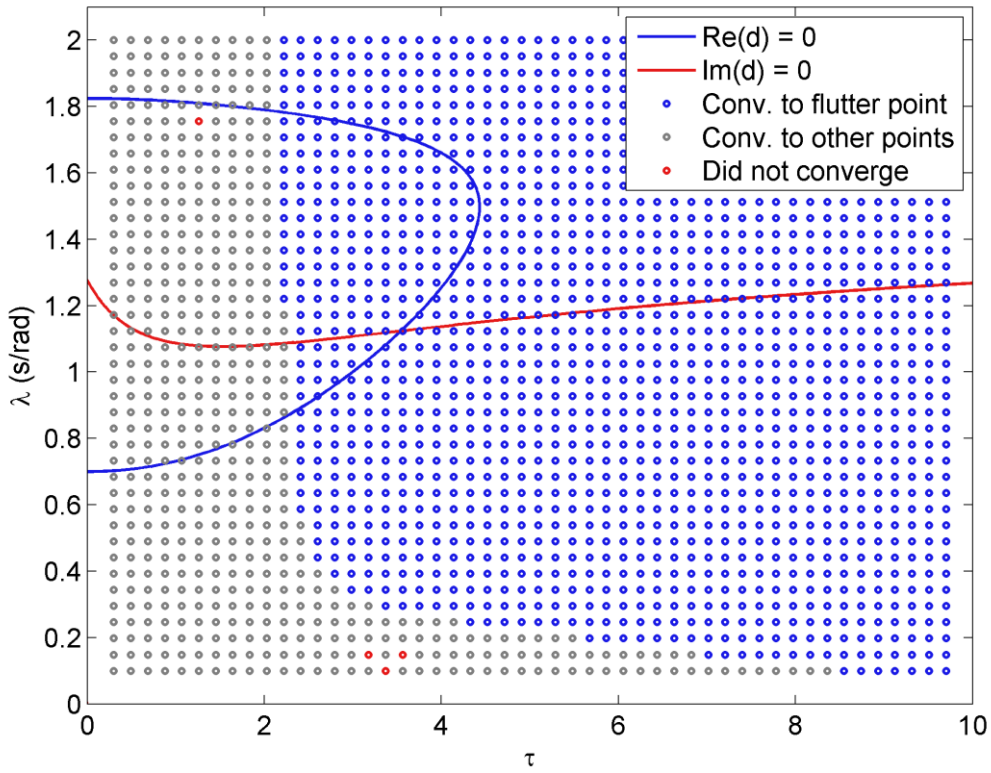


Figure 6.7: Numerical convergence analysis of the SLP algorithm (τ-λ section model).

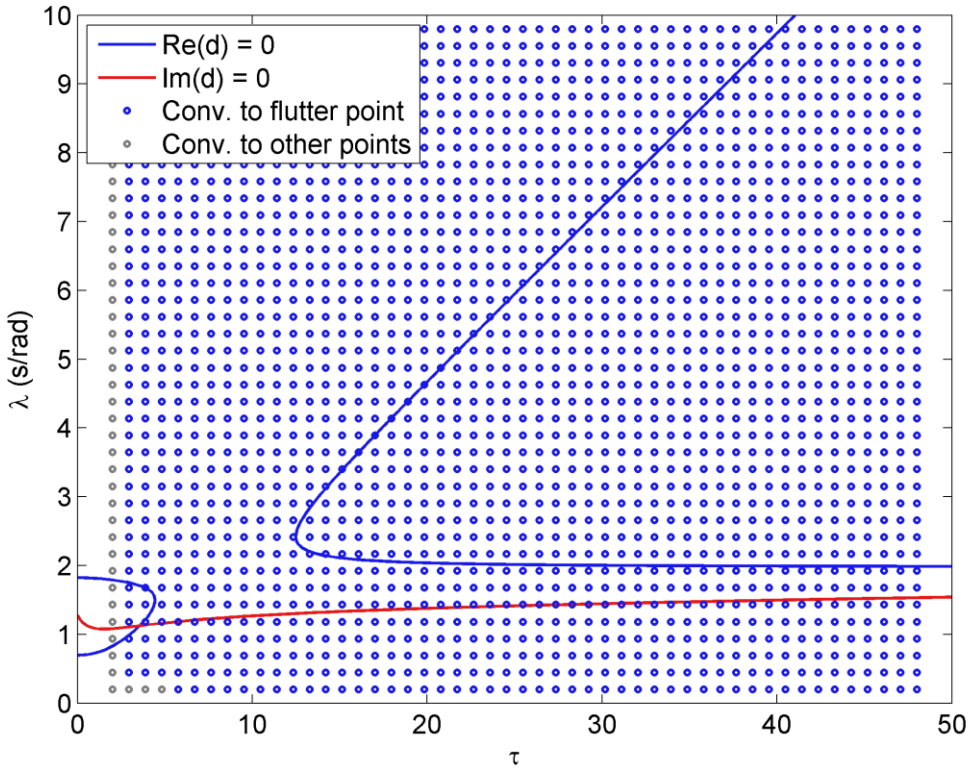


Figure 6.8: Numerical convergence analysis of the SLP algorithm (τ-λ section model, wide field of view).

6.3. THE ITERATED CONTOUR PLOT / NEWTON'S METHOD

6.3.1 Formulation

For some two-parameter nonlinear multiparameter eigenvalue problem $\mathrm{A}(\chi,p)\mathbf{x}=\mathbf{0}$, it is well known that $\det\big(\mathrm{A}(\chi_i,p_i)\big)=0$ must hold for eigenvalues $[\chi_i,p_i]$. In Chapter 3 we introduced the contour plot, based on this idea. Consider the two equations defining the contour plot:

$$\begin{aligned}\mathrm{Re}\big(z(\chi,p)\big)&=0\\ \mathrm{Im}\big(z(\chi,p)\big)&=0,\end{aligned}\qquad(6.3.1)$$

where $z=\det\big(\mathrm{A}(\chi,p)\big)$. Let us approximate these contours with a straight line using information from a single test point, $[\chi_0,p_0]$. To do this we define the plane passing through $[\chi_0,p_0]$ that is tangent to either the surface $\mathrm{Re}\big(z(\chi,p)\big)$ (for the real contour) or the surface $\mathrm{Im}\big(z(\chi,p)\big)$. We then compute the zero-contours of these planes, which are straight lines. Implicitly, the equations of these contours are

$$\begin{aligned}
&\mathrm{Re}\big(z(\chi_0,p_0)\big) + (\chi-\chi_0)\left.\frac{\partial\,\mathrm{Re}(z)}{\partial\chi}\right|_{\chi_0,p_0} + (p-p_0)\left.\frac{\partial\,\mathrm{Re}(z)}{\partial p}\right|_{\chi_0,p_0} = 0\\
&\mathrm{Im}\big(z(\chi_0,p_0)\big) + (\chi-\chi_0)\left.\frac{\partial\,\mathrm{Im}(z)}{\partial\chi}\right|_{\chi_0,p_0} + (p-p_0)\left.\frac{\partial\,\mathrm{Im}(z)}{\partial p}\right|_{\chi_0,p_0} = 0,
\end{aligned} \tag{6.3.2}$$

i.e.

$$\begin{aligned}
&\mathrm{Re}\big(z(\chi_0,p_0)\big) + (\chi-\chi_0)\,\mathrm{Re}\left(\partial_\chi z(\chi_0,p_0)\right) + (p-p_0)\,\mathrm{Re}\left(\partial_p z(\chi_0,p_0)\right) = 0\\
&\mathrm{Im}\big(z(\chi_0,p_0)\big) + (\chi-\chi_0)\,\mathrm{Im}\left(\partial_\chi z(\chi_0,p_0)\right) + (p-p_0)\,\mathrm{Im}\left(\partial_p z(\chi_0,p_0)\right) = 0.
\end{aligned} \tag{6.3.3}$$

Note that Eq. 6.3.2 can be equivalently obtained by taking the first-order Taylor series of Eq. 6.3.1 about $[\chi_0, p_0]$. We are then interested in the intersection of the two contours in Eq. 6.3.3 – the estimated flutter point – which can be computed as

$$\begin{bmatrix}\chi\\ p\end{bmatrix} = \begin{bmatrix}\chi_0\\ p_0\end{bmatrix} - \begin{bmatrix}\mathrm{Re}\left(\partial_\chi z(\chi_0,p_0)\right) & \mathrm{Re}\left(\partial_p z(\chi_0,p_0)\right)\\ \mathrm{Im}\left(\partial_\chi z(\chi_0,p_0)\right) & \mathrm{Im}\left(\partial_p z(\chi_0,p_0)\right)\end{bmatrix}^{-1}\begin{bmatrix}\mathrm{Re}\big(z(\chi_0,p_0)\big)\\ \mathrm{Im}\big(z(\chi_0,p_0)\big)\end{bmatrix}. \tag{6.3.4}$$

The form of Eq. 6.3.4 should be familiar: this is Newton's method. That is to say, if we define a state vector $\mathbf{v} = [\chi, p]^T$ and the complex-valued scalar determinant function $z = \det\big(\mathrm{D}(\mathbf{x})\big)$, we obtain the real-valued system of equations

$$\mathbf{F}(\mathbf{v}) = \begin{bmatrix}\mathrm{Re}\big(z(\mathbf{v})\big)\\ \mathrm{Im}\big(z(\mathbf{v})\big)\end{bmatrix} = \mathbf{0}, \tag{6.3.5}$$

and upon applying Newton's method to this equation we obtain

$$\mathbf{v}_{k+1} = \mathbf{v}_k - \mathrm{J}(\mathbf{v}_k)^{-1}\mathbf{F}(\mathbf{v}_k) \tag{6.3.6}$$

where J is the Jacobian matrix of $\mathbf{F}$, i.e.

$$\mathrm{J}(\mathbf{v}) = \begin{bmatrix} \partial_\chi \operatorname{Re}\big(z(\mathbf{v})\big) & \partial_p \operatorname{Re}\big(z(\mathbf{v})\big) \\ \partial_\chi \operatorname{Im}\big(z(\mathbf{v})\big) & \partial_p \operatorname{Im}\big(z(\mathbf{v})\big) \end{bmatrix} = \begin{bmatrix} \operatorname{Re}\left(\partial_\chi z(\mathbf{v})\right) & \operatorname{Re}\left(\partial_p z(\mathbf{v})\right) \\ \operatorname{Im}\left(\partial_\chi z(\mathbf{v})\right) & \operatorname{Im}\left(\partial_p z(\mathbf{v})\right) \end{bmatrix}. \tag{6.3.7}$$

As can be seen, Eq. 6.3.6 is identical to Eq. 6.3.4. Readers may question our purpose in introducing this method in terms of a linearisation of the contour plot and not simply in terms of Newton's method. We have done this because we will presently extend this method based on ideas from the contour plot, and the motivation for these extensions is not immediately apparent from Newton's method. This form of Newton's method has been applied (in basic form) to two-parameter linear multiparameter systems by Podlevskii [3,8], and (with more subtlety) to general linear multiparameter systems by Dai [9]. The method has not previously been applied to nonlinear or unstructured systems. Other forms of Newton's method have been also applied to more general linear multiparameter systems (though again not to nonlinear systems) [4,10], though these implementations are not equivalent to the iterated contour plot. The iterated contour plot method always iterates on a vector of size 2, irrespective of the size of the original system ($\mathrm{A}(\chi, p)\mathbf{x} = 0$), whereas the multiparameter implementations presented in [10] iterate over a much larger vector, as they include the eigenvector as well as the eigenvalues in this iteration vector. We will discuss other ways of using Newton's method to solve unstructured problems in Section 6.5. Note that it is not difficult to rewrite the iterated contour plot method to apply to systems with greater than two parameters – the Jacobian and residual function definition simply become larger.

The iterated contour plot method compares well to the SLP algorithm of Section 6.2. We should expect the computation time per iteration to be relatively small, as the inverse of the 2×2 Jacobian used in Eq. 6.3.4 can be expressed analytically. However, we would also expect rounding-error problems to occur when the system is large and the determinant is small: it is well known that the determinant does not always provide an accurate measure of system singularity [11,12]. We are fully aware of this weakness and discuss it further in Section 6.9. Note that, like the SLP algorithm, the iterated contour plot method requires the evaluation of derivations. In this case, Eq. 6.3.3 requires the evaluation of the first

derivatives of z with respect to χ and p. These can be computed via finite-differences, but there is also another way of doing it. Jacobi's formula for the derivative of a determinant [13] implies:

$$\frac{\partial z}{\partial x} = \frac{\partial}{\partial x}\det\big(\mathrm{A}(x)\big) = \det(\mathrm{A})\,\mathrm{tr}\left(\mathrm{A}^{-1}\frac{\partial \mathrm{A}}{\partial x}\right) = \mathrm{tr}\left(\mathrm{adj}(\mathrm{A})\frac{\partial \mathrm{A}}{\partial x}\right), \tag{6.3.8}$$

where $\mathrm{adj}(\mathrm{A})$ denotes the adjugate of A. This provides an alternate method of evaluating these derivatives which is useful when A has at least some known structure (and can be differentiated analytically or algorithmically). However, in the general case we will use finite-differences to evaluate $\partial_\chi z$ and other required derivatives directly. Lastly, it is interesting to note that the iterated contour plot algorithm requires no selection step, as the algorithm always produces only one new estimate of the flutter point each iteration (if the Jacobian J is nonsingular). This makes for a simpler algorithm, and means that, for a given system, the convergence properties of an iterative sequence depend only on the associated initial guess. When we analysed the convergence properties of the successive linear problems algorithm there was no such direct link, as a change in the selection algorithm would influence the iterative paths taken by some initial guesses.

One other issue regarding the implementation of iterated contour plot method must be addressed: the iteration does not compute the system's eigenvectors (i.e. the system's fluttering modeshape, in most cases). This is not a great loss from the perspective of the output of the algorithm, as we will often only be interested in the flutter point locations. However, it does mean that we cannot define the residual as per 6.2.10. The Newton-method is based around the minimisation of the residual $\|[\mathrm{Re}(z)\,,\mathrm{Im}(z)\,]^T\|$ i.e. $|\det(\mathrm{A})|$, however, we should be extremely reluctant to use this as the definition of the residual in our algorithm as it will be a poor measure of the system's singularity, as we noted earlier [11,12]. A better definition would be $1/\kappa(\mathrm{A})$, where $\kappa(\mathrm{A})$ denotes the condition number of A. This choice has a reasonable theoretical basis, as $\kappa(\mathrm{A}) = \|\mathrm{A}\|\|\mathrm{A}^{-1}\| = \|\mathrm{A}\|\|\mathrm{adj}(\mathrm{A})\|/\det(\mathrm{A})$ for some given matrix norm and thus we should expect our residual to be roughly proportional to $\det(\mathrm{A})$. However, when the two-norm is used, the condition number can be computed via the singular value decomposition, without recourse to the inverse or determinant.

Algorithm 6.3 – second-order iterated contour plot algorithm

1	**initialise** $k = 0$, initial guesses χ_0, p_0, increments $\delta\chi$, δp and tolerances ϵ_1, ϵ_2
2	evaluate $Z_k = \det\big(\mathrm{A}(\chi_k, p_k)\big)$
3	evaluate $$P_k = \frac{\det\big(\mathrm{A}(\chi_k, p_k + \delta p)\big) - Z_k}{\delta p}$$ $$X_k = \frac{\det\big(\mathrm{A}(\chi_k + \delta\chi, p_k)\big) - Z_k}{\delta\chi}$$
4	compute the solution $[\Delta\chi_k, \Delta p_k]^T$ of the linear system $$\begin{bmatrix} \mathrm{Re}(X_k) & \mathrm{Re}(P_k) \\ \mathrm{Im}(X_k) & \mathrm{Im}(P_k) \end{bmatrix} \begin{bmatrix} \Delta\chi_k \\ \Delta p_k \end{bmatrix} = \begin{bmatrix} \mathrm{Re}(Z_k) \\ \mathrm{Im}(Z_k) \end{bmatrix}$$
5	evaluate $\chi_{k+1} = \Delta\chi_k + \chi_k$, $p_{k+1} = \Delta p_k + p_k$
6	**if** $\Delta\chi_k^2 + \Delta p_k^2 < \epsilon_1$ and $\kappa\big(\mathrm{A}(\chi_{k+1}, p_{k+1})\big) < \epsilon_2$
7	**then** return p_k, χ_k
8	**else** $k = k + 1$, **goto** (2)

6.3.2 Numerical experiments

Figure 6.9 shows a contour plot with nine different iteration paths of the first-order iterated contour plot (from nine different initial guesses) superimposed. All of the iteration paths converge: three to the first flutter point, one to the divergence point and five to the other flutter points. There is again an exact agreement between the converged values from the algorithm and the flutter points predicted by the contour plot. Figure 6.10 shows a logarithmic convergence plot for an initial guess of $\chi_0 = 1\ \mathrm{rad/s}$, $\Upsilon_0 = 5\ \mathrm{Hz}$. This iterative sequence converges to the first flutter point, with uniform convergence as can be seen. The gradient of this plot (order of convergence) can be measured as 1.91, confirming our convergence analysis that the method is has quadratic convergence. Figure 6.11 shows a logarithmic convergence plot for an initial guess of $\chi_0 = 1\ \mathrm{rad/s}$, $\Upsilon_0 = 5\ \mathrm{Hz}$. This iterative sequence converges to the divergence point. Note the difference in uniformity between Figure 6.10 and Figure 6.11: the convergence to the divergence point is less uniform. It also has lower order: the average gradient of Figure 6.11 is approximately 1.03.

Figures 6.12 and 6.13 show numerical convergence analyses of the iterated contour plot algorithm (applied to the section model with Theodorsen aerodynamics), of the same form as Figure 6.4. Overall, the convergence behaviour of the algorithm is good – there is a wide basin of attraction to the first flutter point. This basin extends further into the upper-right quarter plane, as can be seen in Figure 6.13, and again it is possible that this is unbounded. There is a clear boundary lying approximately on the line $\chi = 1 + 0.23\Upsilon$. However, the behaviour outside this boundary is somewhat chaotic, with a thin line of points converging to the divergence point, and then a mix of points converging to other flutter points and non-converging points, with one or two points converging to the first flutter point or divergence point. This effect is unchanged when we increase the maximum permitted number of iterations in the algorithm settings, and so one cannot interpret the non-converging points simply as points that would converge given more iterations. However, we might link this effect with the fact that Newton's iteration is known to produce fractal behaviour when iterating over the complex plane [14]. While we are not actually iterating with complex-valued variables, we are dealing with a complex-valued matrix function. This is an area for future investigation.

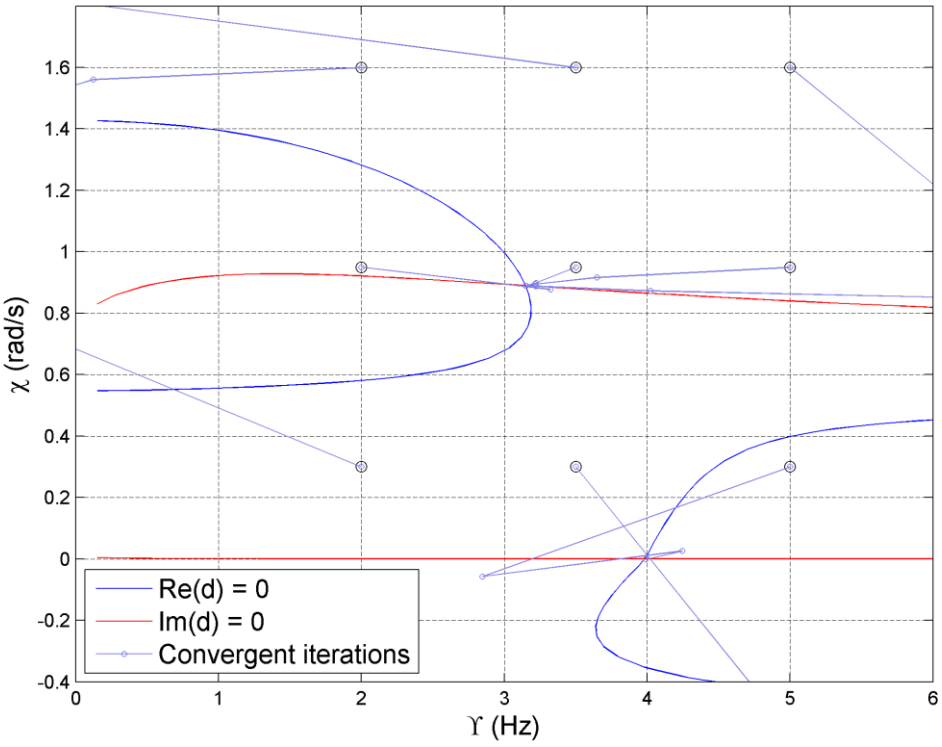


Figure 6.9: Contour plot with nine different iteration paths of the first-order iterated contour plot method.

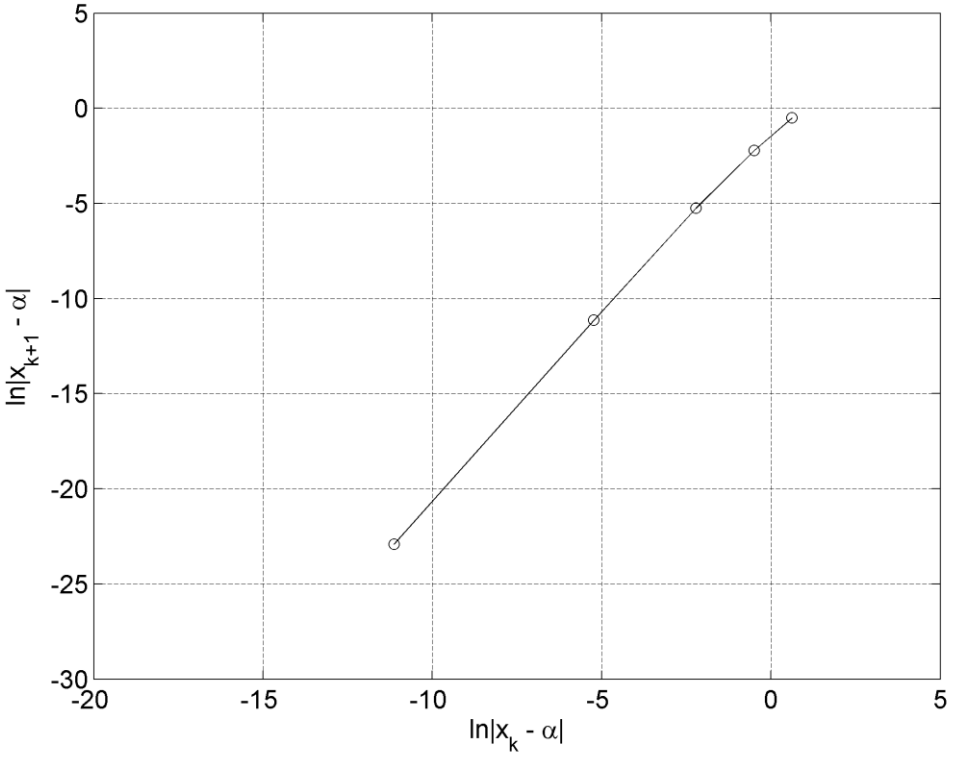


Figure 6.10: Logarithmic convergence plot for the iterated contour plot algorithm converging to the flutter point ($\chi_0 = 1\ \mathrm{rad/s}, \Upsilon_0 = 5\ \mathrm{Hz}$).

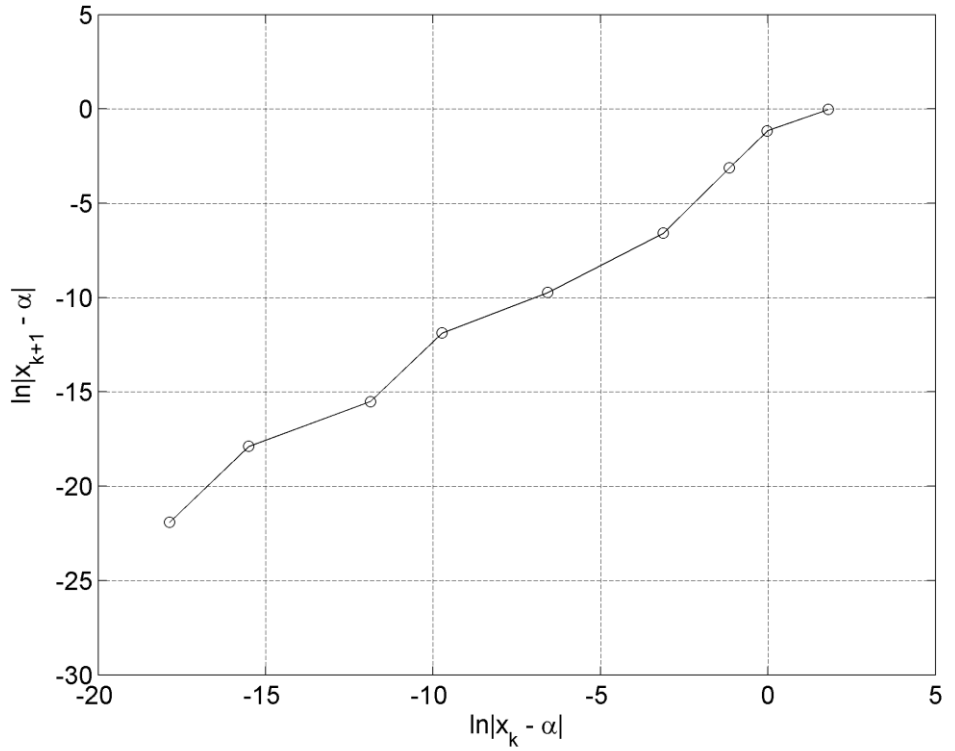


Figure 6.11: Logarithmic convergence plot for the iterated contour plot algorithm converging to the divergence point ($\chi_0 = 0.3\ \mathrm{rad/s}, \Upsilon_0 = 10\ \mathrm{Hz}$).

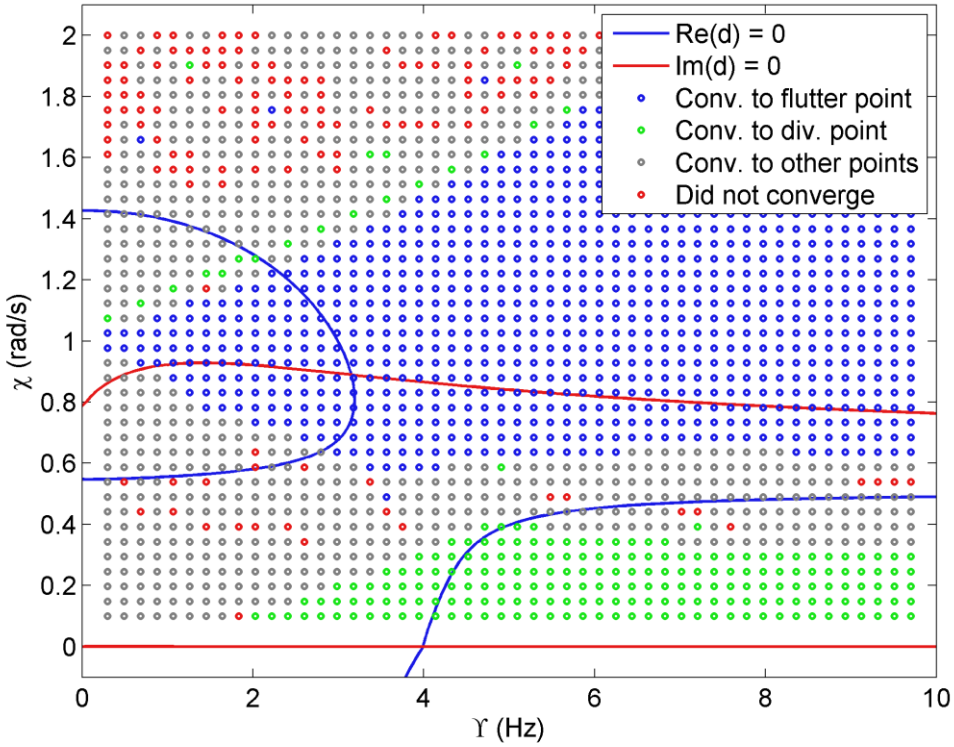


Figure 6.12: Numerical convergence analysis of the iterated contour plot method (Υ-χ section model)

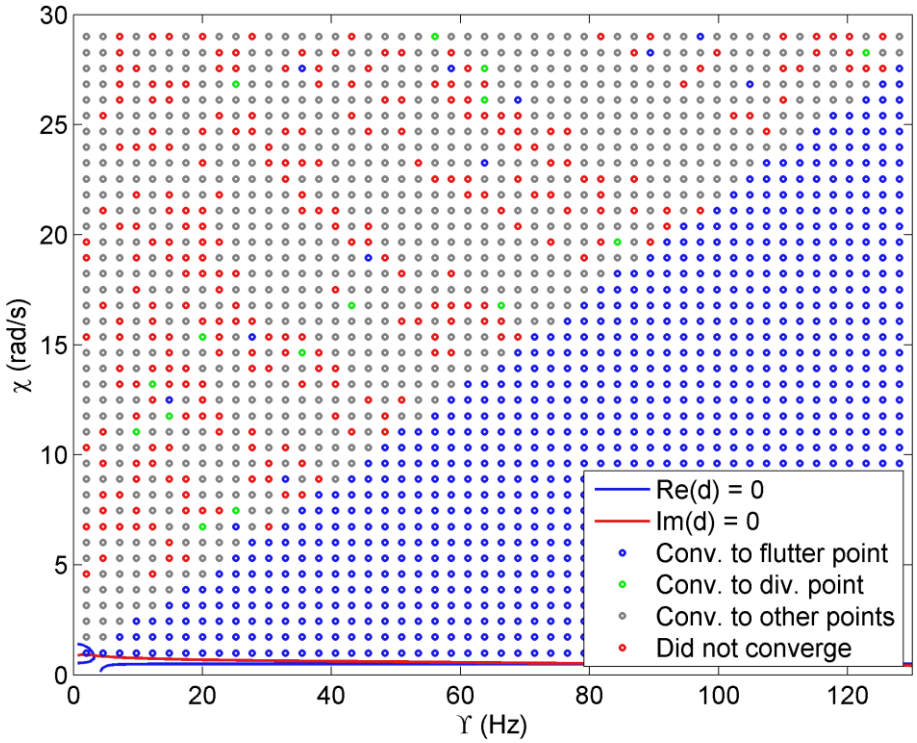


Figure 6.13: Numerical convergence analysis of the iterated contour plot method (Υ-χ section model, wide field of view).

6.4. HIGHER-ORDER METHODS

6.4.1 Motivation

We have thus far considered methods based on first-order approximations of the unstructured multiparameter eigenvalue problem. Following the same line of reasoning as we pursued in Chapter 5, we might wish to consider higher-order approximations, as this could give us more global convergence, and more flexibility in choosing the flutter point we wish to converge to. Polynomial or rational approximations are an obviously good choice, as we have presented the framework in Chapter 4 whereby any polynomial multiparameter eigenvalue problem can be solved directly. Computational expense and the size of the operator determinants is a limiting factor, however. In Chapter 2 we replaced Theodorsen's function with a rational expression and solved it with a direct solver. There, however, we were dealing with a known function, and we had the leisure to try out different approximations of this function and compare their effectiveness. Perhaps more importantly, we also had an existing body of literature regarding that function and its approximation. When we are dealing with a general and potentially unknown matrix functions, we are not so fortunate. It is reasonable, therefore, to start improving our approximation in simple steps. One approximation which we could make is the introduction of higher-order terms in the Taylor expansions used in the SLP or iterated contour plot algorithms.

As a first development in this regard, we will consider increasing the order of accuracy of the Taylor approximation in our iterated contour plot algorithm. This corresponds to approximating the contour plot with a second order algebraic curve – a conic section. Note that if we had introduced the iterated contour plot algorithm simply as Newton's method, the physical interpretation of increasing the order of the Taylor series approximation would not be apparent.

6.4.2 A second-order iterated contour plot method

If we include second order terms in the multivariate Taylor series of Eq. 6.3.2, we obtain the following approximate contour plot, which takes the form of a conic section. This plot may be expressed as

$$\begin{aligned}
&\mathrm{Op}\big(z(\chi_0,p_0)\big) + (\chi-\chi_0)\,\mathrm{Op}\left(\left.\frac{\partial z}{\partial \chi}\right|_{\chi_0,p_0}\right) + (p-p_0)\,\mathrm{Op}\left(\left.\frac{\partial z}{\partial p}\right|_{\chi_0,p_0}\right) \\
&\quad + (\chi-\chi_0)(p-p_0)\frac{1}{2}\mathrm{Op}\left(\left.\frac{\partial z}{\partial \chi \partial p}\right|_{\chi_0,p_0}\right) \\
&\quad + (\chi-\chi_0)^2\frac{1}{2}\mathrm{Op}\left(\left.\frac{\partial^2 z}{\partial \chi^2}\right|_{\chi_0,p_0}\right) + (p-p_0)^2\frac{1}{2}\mathrm{Op}\left(\left.\frac{\partial^2 z}{\partial p^2}\right|_{\chi_0,p_0}\right) = 0,
\end{aligned} \tag{6.4.1}$$

where we use a generic operator, $\mathrm{Op}(\cdot) \in \{\mathrm{Re}(\cdot), \mathrm{Im}(\cdot)\}$, instead of writing the equation twice with each of these operators. Eq. 6.4.1 may be expressed more simply as

$$\begin{aligned}
R_0 + \Delta\chi R_\chi + \Delta p R_p + \Delta\chi\Delta p R_{\chi p} + \Delta\chi^2 R_{\chi\chi} + p^2 R_{pp} = 0 \\
C_0 + \Delta\chi C_\chi + \Delta p C_p + \Delta\chi\Delta p C_{\chi p} + \Delta\chi^2 C_{\chi\chi} + p^2 C_{pp} = 0
\end{aligned} \tag{6.4.2}$$

with $\Delta\chi = \chi - \chi_0$, $\Delta p = p - p_0$. The coefficients of Eq.. 6.4.2 are easy to infer: we define them fully in terms of finite differences in Algorithm 6.5. Multiplying Eq. 6.4.2 by arbitrary scalar eigenvector parameters x and y, we have nothing other than a quadratic multiparameter eigenvalue problem in $\Delta\chi$ and Δp, with scalar coefficients. We can solve this via the direct solvers described in Chapter 4. We then have to select $(\Delta\chi, \Delta p)$ from the four eigenvalue pairs of Eq. 6.4.2. We can eliminate those eigenvalue pairs with a complex component, but it will still usually be necessary to choose from more than one real eigenvalue pair. We thus have to use one of the selection algorithms described in Chapter 5. The most sensible of these is the selector based on minimising of the Euclidean distance between the next iterate ($\chi_{k+1} = \chi_k + \Delta\chi$, $p_{k+1} = p_k + \Delta p$), and the current iterate (χ_k, p_k). The Euclidean distance (squared) between the iterates is thus obviously $\Delta\chi_k^2 + \Delta p_k^2$, and we choose $(\Delta\chi, \Delta p) \in \mathbb{R}^2$ such that this is minimised. It may arise that Eq. 6.4.2 has no real eigenvalues (only complex ones): in this case the iteration must be terminated and run from a different starting point (or with a different selection algorithm).

Algorithm 6.4 presents the second-order iterated contour plot algorithm. Algorithm 6.5 presents an efficient method of evaluating the 12 derivative parameters required in Eq. 6.4.2. This method uses second-order central differences and is constructed in such a way as to minimise the number of matrix-function evaluations – $\mathrm{A}(\chi, p)$ or $\det\big(\mathrm{A}(\chi, p)\big)$ – which is

likely to be the most computationally-intensive component in the code when A is large or defined by its own intensive algorithm.

Algorithm 6.4 – second-order iterated contour plot algorithm

1	**initialise** $k = 0$, initial guesses χ_0, p_0, increments $\delta\chi$, δp and tolerances ϵ_1, ϵ_2
2	evaluate R_0, R_χ, R_p, $R_{\chi p}$, $R_{\chi\chi}$, R_{pp}, C_0, C_χ, C_p, $C_{\chi p}$, $C_{\chi\chi}$, C_{pp} with increments $\delta\chi$, δp and location $[\chi_k, p_k]$ from Algorithm 6.5
3	compute the set of solutions $\{\Delta\chi_{k,(i)}\}$, and $\{\Delta p_{k+1,(i)}\}$ to $$\left(R_0 + \Delta\chi_k R_\chi + \Delta p_k R_p + \Delta\chi_k \Delta p_k R_{\chi p} + \Delta\chi_k^2 R_{\chi\chi} + p_k^2 R_{pp}\right)x = 0$$ $$\left(C_0 + \Delta\chi_k C_\chi + \Delta p_k C_p + \Delta\chi_k \Delta p_k C_{\chi p} + \Delta\chi_k^2 C_{\chi\chi} + p_k^2 C_{pp}\right)y = 0$$
4	select eigenvalues $\Delta p_k \in \{\Delta p_{k,(i)}\}$, $\Delta\chi_k \in \{\Delta\chi_{k,(i)}\}$ such that $\Delta\chi_k^2 + \Delta p_k^2$ is minimised
5	evaluate $\chi_{k+1} = \Delta\chi_k + \chi_k$, $p_{k+1} = \Delta p_k + p_k$
6	**if** $\Delta\chi_k^2 + \Delta p_k^2 < \epsilon_1$ and $\kappa\left(\mathrm{A}(\chi_{k+1}, p_{k+1})\right) < \epsilon_2$
7	**then** return p_k, χ_k and $\mathbf{x}_k$
8	**else** $k = k + 1$, **goto** (2)

Algorithm 6.5 – second-order iterated contour plot algorithm

1	**initialise** location $[\chi_0, p_0]$ and increments $\delta\chi$, δp
2	evaluate $D_{0,0} = \det\left(\mathrm{A}(\chi_0, p_0)\right)$, $D_{1,0} = \det\left(\mathrm{A}(\chi_0 + \delta\chi, p_0)\right)$, $D_{-1,0} = \det\left(\mathrm{A}(\chi_0 - \delta\chi, p_0)\right)$, $D_{0,-1} = \det\left(\mathrm{A}(\chi_0, p_0 - \delta p)\right)$, $D_{0,1} = \det\left(\mathrm{A}(\chi_0, p_0 + \delta p)\right)$, $D_{1,1} = \det\left(\mathrm{A}(\chi_0 + \delta\chi, p_0 + \delta\chi)\right)$, $D_{2,0} = \det\left(\mathrm{A}(\chi_0 + 2\delta\chi, p_0)\right)$ and $D_{0,2} = \det\left(\mathrm{A}(\chi_0, p_0 + 2\delta p)\right)$
4	evaluate $$D_\chi = \frac{D_{1,0} + D_{-1,0}}{2\delta\chi}$$ $$D_p = \frac{D_{0,1} + D_{0,-1}}{2\delta p}$$ $$D_{\chi p} = \frac{D_{2,0} - D_{1,0} - D_{0,1} + 2D_{0,0} - D_{-1,0} - D_{0,-1} + D_{0,2}}{2\delta\chi\delta p}$$ $$D_{\chi\chi} = \frac{D_{1,0} - 2D_{0,0} + D_{-1,0}}{\delta\chi^2}$$

	$$D_p = \frac{D_{0,1} - 2D_{0,0} + D_{0,-1}}{\delta p^2}$$
5	evaluate $R_0 = \mathrm{Re}(D_{0,0}),\ C_0 = \mathrm{Im}(D_{0,0}),\ R_\chi = \mathrm{Re}(D_\chi),\ C_\chi = \mathrm{Im}(D_\chi)$ $R_{\chi p} = \frac{1}{2}\mathrm{Re}(D_{\chi p}),\ C_{\chi p} = \frac{1}{2}\mathrm{Im}(D_{\chi p}),\ R_{\chi\chi} = \frac{1}{2}\mathrm{Re}(D_{\chi\chi}),\ C_{\chi\chi} = \frac{1}{2}\mathrm{Im}(D_{\chi\chi})$ $R_{pp} = \frac{1}{2}\mathrm{Re}(D_{pp}),\ C_{pp} = \frac{1}{2}\mathrm{Im}(D_{pp})$
6	return R_0, R_χ, R_p, $R_{\chi p}$, $R_{\chi\chi}$, R_{pp}, C_0, C_χ, C_p, $C_{\chi p}$, $C_{\chi\chi}$, C_{pp}

6.4.3 Numerical experiments

We again simulate the nondimensional section model with Theodorsen aerodynamics, in χ-Υ form. Parameter values are taken from Chapter 2. Figure 6.14 shows a contour plot with nine different iteration paths (from nine different initial guesses) superimposed. Five of the iteration paths converge to the first flutter point, one to flutter points with negative airspeed values, two to the divergence point, and one iteration does not converge. The initial guesses that do not converge are unable to even supply one iterate – all the solutions to Eq. 6.4.2 at these initial guesses are complex. An excellent agreement is observed between Algorithm 6.4's estimates for the flutter and divergence points, and those of the contour plot.

Figure 6.15 shows a logarithmic convergence plot for an initial guess of $\chi_0 = 1\ \mathrm{rad/s}$, $\Upsilon_0 = 5\ \mathrm{Hz}$. The iterative sequence converges to the first flutter point. The convergence rate of this iteration is relatively uniform at second order. Figure 6.16 shows a logarithmic convergence plot for an initial guess of $\chi_0 = 0.3\ \mathrm{rad/s}$, $\Upsilon_0 = 10\ \mathrm{Hz}$. The iterative sequence converges to the divergence point and, as expected, the convergence decreases to first-order.

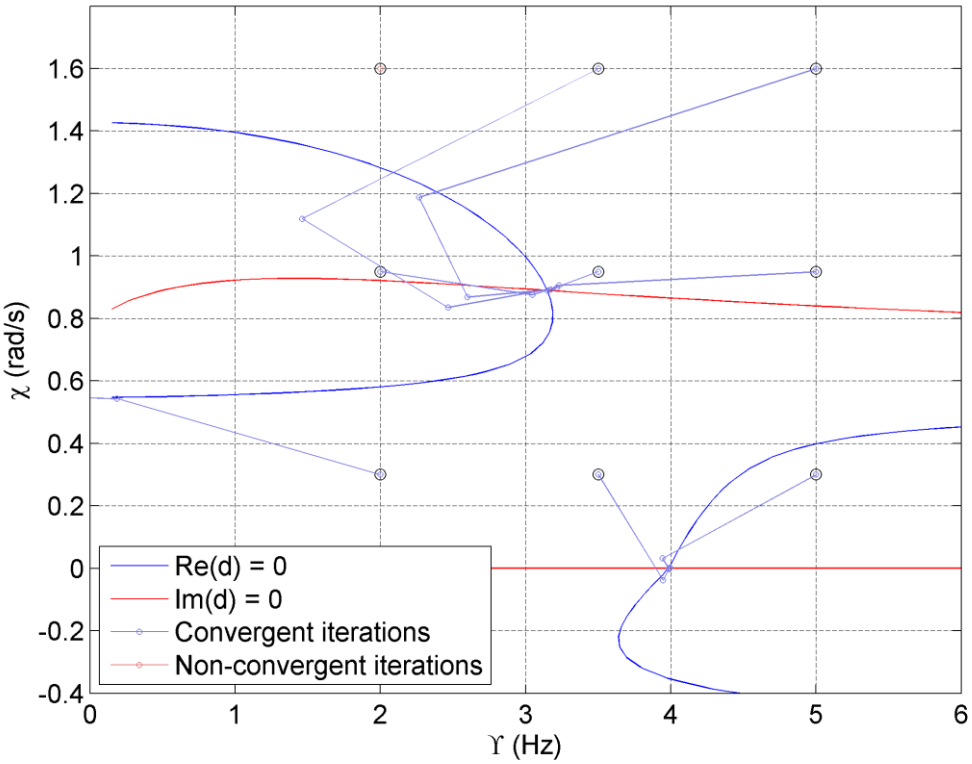


Figure 6.14: Contour plot with nine iteration paths for the second-order iterated contour plot method.

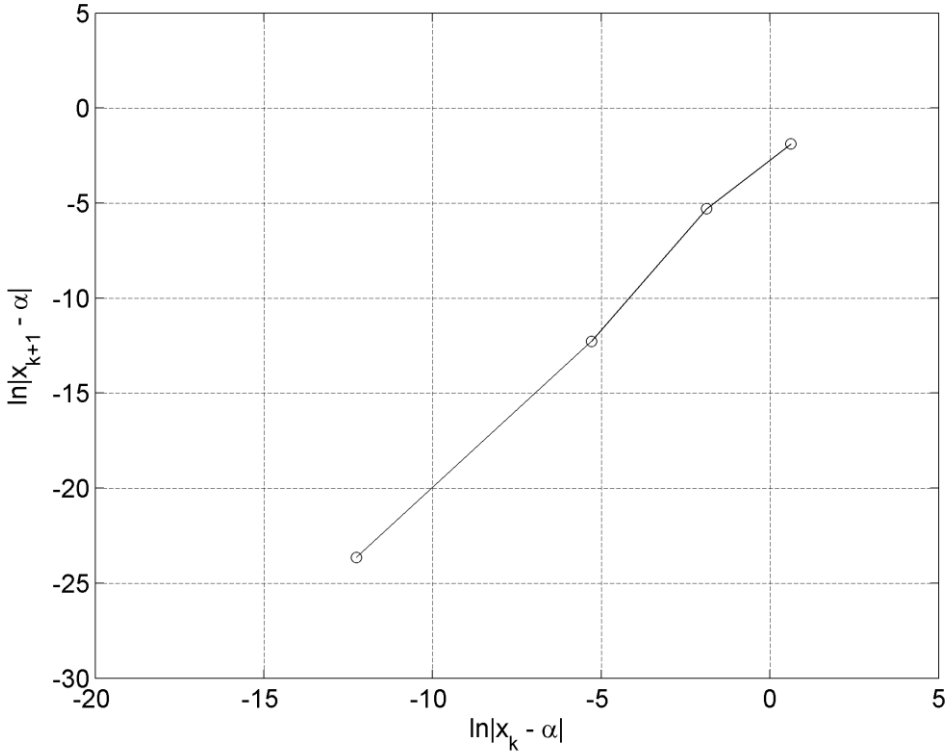


Figure 6.15: Logarithmic convergence plot for the second-order iterated contour plot algorithm converging to the flutter point ($\chi_0 = 1$ rad/s, $\Upsilon_0 = 5$ Hz).

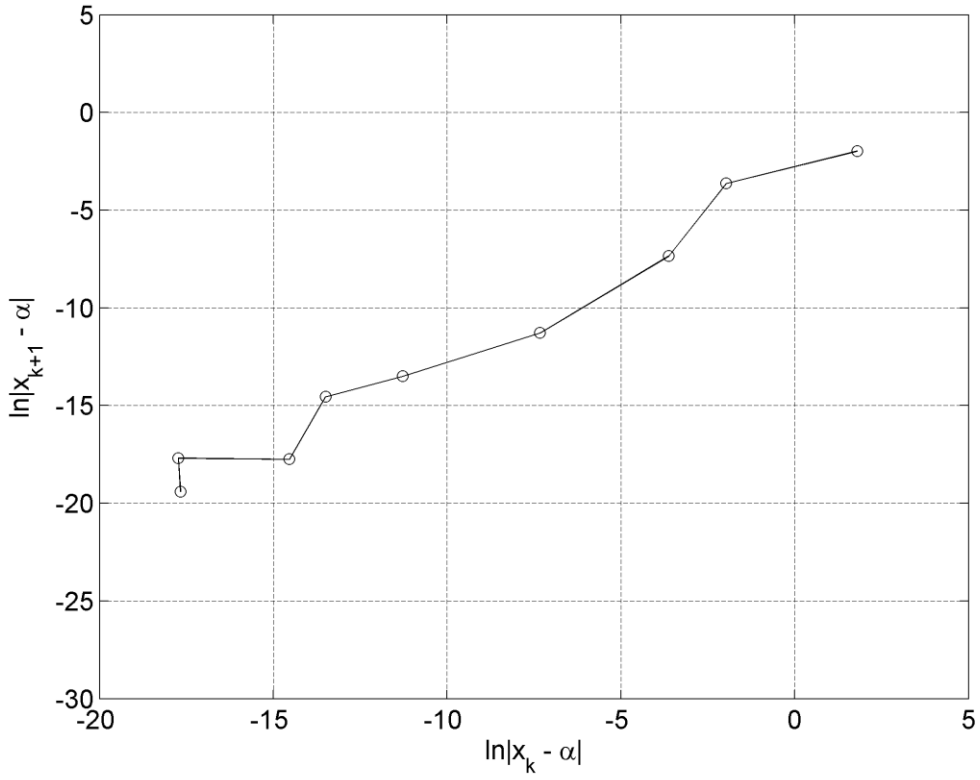


Figure 6.16: Logarithmic convergence plot for the second-order iterated contour plot algorithm converging to the divergence point ($\chi_0 = 0.3\ \mathrm{rad/s}, \Upsilon_0 = 10\ \mathrm{Hz}$).

Figure 6.17 shows the results of a numerical convergence analysis performed in the vicinity of the first flutter point. As can be seen, there is an enclosed basin of attraction around the first flutter point, in which the iterates converge to this flutter point. Below this basin there is another (centred around the divergence point), and above it the iterates do not converge (most produce only complex solutions at the first iteration). To the left of the first flutter point basin, there are scattered areas of convergence to flutter points below $\Upsilon < 0$, mixed with areas of no convergence. Comparing Figure 6.12 to Figure 6.17 we can see that the first-order method actually performs significantly better than the second-order variant: the convergence speed is not improved by increasing the order of convergence, but the basin of attraction around the first flutter point contracts massively. Unlike in Figure 6.12, in Figure 6.17 there is a large area above the first flutter point (in red) where the iterations do not converge to any flutter point – most of them failing to produce even one iteration. This is the disadvantage of the second-order method: because the iteration cannot proceed when only complex roots for Eq. 6.4.2 are generated, the method is prone to non-convergence.

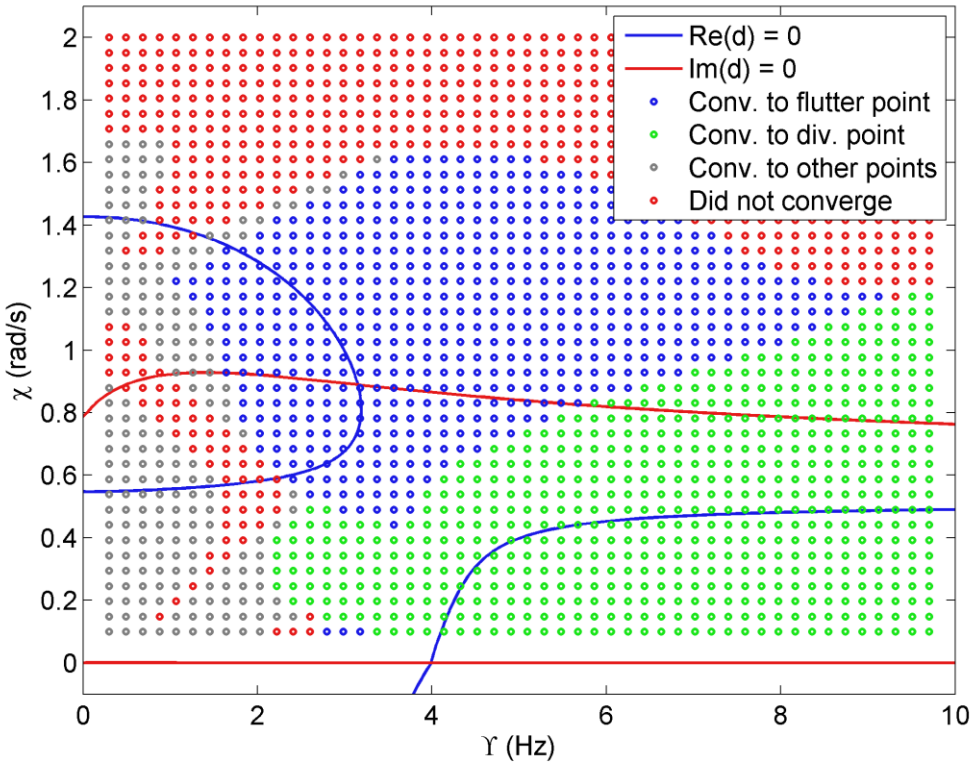


Figure 6.17: Numerical convergence analysis of the second-order iterated contour plot method (Υ-χ section model).

6.4.4 Retrospective review

The second-order contour plot performs worse than the first-order equivalent in terms of the size of the basins of attraction round each physical flutter point, and equal in terms of convergence order. This is not a positive result. The lack of large converge basins is a crippling disadvantage, especially as we were primarily interested in this method because of its potential to provide better global convergence properties. We could put a safeguard against this behaviour by switching to the first-order iterated contour plot when the second-order method does not produce a suitable iterate. However, this begs the question of why we would use the second-order method at all, given that it has not shown itself to be faster than the first-order method when it does converge.

In retrospect, we could have realised that it was not a particularly wise choice to increase the determinant Taylor expansion order of the iterated contour plot to two, and that such a move would probably not bring any immediate benefits. The matrix function governing the section model with Theodorsen aerodynamics is

$$\mathrm{A}(\chi,\Upsilon) = (\mathrm{M}_0 + \mathrm{G}_2)\chi^2 + \left(\mathrm{G}_1 + \mathrm{G}_2 C\left(\frac{\chi}{\Upsilon}\right)\right)\Upsilon\chi + \mathrm{G}_3 C\left(\frac{\chi}{\Upsilon}\right)\Upsilon^2 - \mathrm{D}_0\chi - \mathrm{K}_0. \qquad (6.4.3)$$

Given that the matrices M_0 etc. are of size 2×2, the determinant of A will be a fourth-order polynomial in Υ and χ, excluding the nonlinearity of Theodorsen's function. Therefore, to get an approximation that would represent Eq. 6.4.3 to a good level of accuracy, we would probably have to increase the order of the determinant Taylor expansion to four. This would require the evaluation of 14 different derivatives. Of course, not all aeroelastic problems will be of a similar form to Eq. 6.4.3, but in general we would expect models of aerospace structures to have quadratic dependence on modal frequency (arising from ubiquitous spring-mass-damper model) and a quadratic dependence on airspeed (given the well-established drag-coefficient / lift-coefficient empirical equations).
It is clear, therefore, that increasing the order of accuracy of the contour plot approximation is not a feasible way of obtaining a method that is likely to have good global-convergence properties for aeroelastic systems: the expansion order required is simply too large.

However, there is slightly more hope if we look at increasing the Taylor expansion order used in the SLP algorithm. For the SLP algorithm we take the Taylor expansion of the matrix function itself, and not its determinant. We expect the matrix function to be quadratic in airspeed and modal frequency, and, whatever order it is, it will remain so irrespective of the size of the coefficient matrices. Therefore, if we increase the Taylor expansion order to two, we should without further manipulation obtain a good approximation to the expected matrix function. This is a great opportunity for future unstructured solver development.

6.5. OTHER NEWTON-TYPE METHODS

Several other methods based on the Newton iterative procedure are known for nonlinear one-parameter eigenvalue problems. A sample of these methods may be found in [5], and many of them are straightforward to extend to the multiparameter case – though due to constraints of size we cannot perform numerical experiments on all of them. Here we give a brief overview of these methods, as applied to multiparameter problems.

6.5.1 Iteration based on minimum singular value

At any flutter point of the problem $\mathrm{A}(\chi, p)\mathbf{x} = \mathbf{0}$, one of the singular values of the matrix A must be zero. We can thus reformulate the eigenvalue problem into a minimisation problem, the objective being to minimise the minimum singular value (σ_1) of A, subject to the constraint $\sigma_1 = 0$. Note that we do not required the eigenvector to evaluate σ_1, and so we need only iterate over the eigenvalue space. We can evaluate σ_1 either by computing the singular value decomposition of A or by performing an inverse iteration on the Hermitian conjugate of A multiplied by A ($\mathrm{A}^H\mathrm{A}$) and taking the square root of the resulting converged eigenvalue[1]. This minimisation problem can of be treated with Newton's method by reformulating it again into the problem of finding the roots of the gradient of σ_1 (i.e. solving $\nabla\sigma_1 = 0$). Newton's method can then be applied to the residual function $\nabla\sigma_1$. As far as we know this method has not been presented before for even one-parameter problems, but it is not particularly novel and does not have much to recommend it.

There are of course other many methods of solving the minimisation problem, and these are perhaps more interesting. Other iterative algorithms, such as the conjugate gradient methods, may be most appropriate to problems with simpler flutter behaviour; for larger problems with more complex behaviour (discontinuities and the like) a simulated annealing or particle swarm optimisation may be attractive. Treating the flutter problem as a minimisation problem does appear to have some potential, particularly when dealing with large and complex problems, where particle swarm optimisation (or a related heuristic method) has the potential to provide near-global convergence, something that is difficult to obtain with standard iterative methods. However, it should be noted that the minimisation approach does have the disadvantage of introducing spurious solutions (minima of σ_1 where $\sigma_1 \neq 0$) which will attract the minimisation algorithm but are not flutter points. If there are many such minima then the real flutter points will be difficult to locate, and the problem will probably have to be treated with a particle-swarm method. But there may be ways to modify the residual definition to eliminate the nonzero flutter points, and this is still a promising avenue for future research. When the flutter problem is defined in more than two

[1] Inverse iteration (with zero shift) being well-known to converge to the smallest eigenvalue of a system [7]. Note that the singular values of A are the square root of the eigenvalues of $\mathrm{A}^H\mathrm{A}$ [11,15]

dimensions, with more than one matrix function (i.e. not simply A and $\overline{\mathrm{A}}$, but a set of A_i) then it is possible to compute the flutter points by minimising the sum of the minimum singular values of each function ($\sum \sigma_{1,i}$). Because the singular values are by definition always real and non-negative, the case $\sigma_{1,i} = 0 \; \forall i$ (a flutter point) will always correspond to a global minimum. Spurious flutter points at local minima will still exist.

A possible alternative approach to problems with more than one matrix function is to define a vector residual containing the minimum singular values of each matrix function ($\mathbf{F}(\mathbf{v}) = \left[\sigma_{1,1}, \sigma_{1,2}, \sigma_{1,3}, \dots\right]$). This has the advantage that no spurious flutter points are introduced. However, it requires that each matrix function be distinct, else the Jacobian of the residual function will be singular. As such it is not applicable to the flutter problems that we have been focusing on (i.e. systems with only A and $\overline{\mathrm{A}}$), but it remains a useful possibility for more general systems.

6.5.2 Full eigenvalue / eigenvector iteration

Consider first the general two-parameter unstructured problem:

$$\begin{aligned} \mathrm{A}_1(\chi, p)\mathbf{x}_1 &= \mathbf{0} \\ \mathrm{A}_2(\chi, p)\mathbf{x}_2 &= \mathbf{0}. \end{aligned} \tag{6.5.1}$$

We observe that these two equation are already in the form of a vector residual function $\mathbf{F}(\mathbf{v}) = \mathbf{0}$. However, to evaluate these residual functions we require the eigenvectors $\mathbf{x}_i$. When we devised the SLP algorithm in Section 6.2 we avoided this problem by linearising the matrix function and solving it with a direct solver (allowing us to advance the iteration without having to compute the residual, except as part of convergence criterion after the iteration was complete).

Alternately, we could simply include the eigenvectors $\mathbf{x}_i$ in our iteration vector, and define $\mathbf{v} = [\chi; p; \mathbf{x}_1; \mathbf{x}_2]$. However, our iteration vector is now larger than the combined residual function ($\mathbf{F}(\mathbf{v}) = [\mathrm{A}_1(\chi, p)\mathbf{x}_1, \mathrm{A}_2(\chi, p)\mathbf{x}_2]$). Two elements are missing. The under-constrained nature of this system makes perfect physical sense, as when the eigenvalues are at the flutter point the eigenvectors can be scaled arbitrarily. We must define the scaling, and a convenient way of doing this is to apply a normalisation. We then define two

extra elements of the residual, $\mathbf{x}_1^H\mathbf{x}_1 - 1 = 0$ and $\mathbf{x}_2^H\mathbf{x}_2 - 1 = 0$. The residual function and iteration vector are now

$$\mathbf{F}(\mathbf{v}) = \begin{bmatrix} \mathrm{A}_1(\chi, p)\mathbf{x}_1 \\ \mathrm{A}_2(\chi, p)\mathbf{x}_2 \\ \mathbf{x}_1^H\mathbf{x}_1 - 1 \\ \mathbf{x}_2^H\mathbf{x}_2 - 1 \end{bmatrix}, \qquad \mathbf{v} = \begin{bmatrix} \chi \\ p \\ \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}. \tag{6.5.2}$$

Newton's method can then be applied directly to Eq. 6.5.2. Note that it is easy to generalise Eq. 6.5.2 to the m-parameter case (with vector of eigenvalues $\boldsymbol{\lambda}$): we simply have

$$\mathbf{F}(\mathbf{v}) = \begin{bmatrix} \mathrm{A}_1(\boldsymbol{\lambda})\mathbf{x}_1 \\ \vdots \\ \mathrm{A}_m(\boldsymbol{\lambda})\mathbf{x}_m \\ \mathbf{x}_1^H\mathbf{x}_1 - 1 \\ \vdots \\ \mathbf{x}_m^H\mathbf{x}_m - 1 \end{bmatrix}, \qquad \mathbf{v} = \begin{bmatrix} \boldsymbol{\lambda} \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_2 \end{bmatrix}. \tag{6.5.3}$$

However, we find that when we are dealing with the two-parameter flutter problems that we consider in this work, Eq. 6.5.2 is somewhat redundant, as we have $\mathrm{A}_2 = \overline{\mathrm{A}}_1$ and $\mathbf{x}_2 = \bar{\mathbf{x}}_1$. The two eigenproblem residuals $\mathrm{A}(\chi, p)\mathbf{x}$ and $\overline{\mathrm{A}}(\chi, p)\bar{\mathbf{x}}$ will give residuals that are simply conjugate to each other, and the two normalisation equations are equivalent, as $\overline{\bar{\mathbf{x}}^H\bar{\mathbf{x}}} = \overline{\bar{\mathbf{x}}^H}\bar{\bar{\mathbf{x}}} = \bar{\mathbf{x}}^T\mathbf{x} = \mathbf{x}^H\mathbf{x}$, but $\mathbf{x}^H\mathbf{x}$ is always real, and so $\bar{\mathbf{x}}^H\bar{\mathbf{x}} = \mathbf{x}^H\mathbf{x}$. The fact that the eigenproblem residuals are conjugate to each other will probably not cause the Newton iteration to fail, but the fact that the normalisation conditions are the same will do so, as the Jacobian will be singular. Note that we should be very careful about describing the two parameter problem (Eq. 6.5.1) with $\mathrm{A}_2 = \overline{\mathrm{A}}_1$ and $\mathbf{x}_2 = \bar{\mathbf{x}}_1$ as being redundant or overconstrained – as we have seen in Chapter 4, the second equation is entirely necessary when using a direct solver. It is only in the context of this method that we can describe the second equation as redundant, and the reasons behind this are not fully understood. We discuss this issue of constrainedness in Chapter 7.

If we simply remove the second eigenvector normalisation condition and the second eigenproblem residual from the residual definition (and the conjugate eigenvector from the iteration variable), the problem then becomes underconstrained, as we have one more element in the iteration variable than in the residual definition. One possibility of

circumventing this problem is to reformulate the two real eigenvalues into one complex-valued eigenvalue $q = \chi + \iota p$. This yields

$$\mathbf{F}(\mathbf{v}) = \begin{bmatrix} \mathrm{A}(\mathrm{Re}(q), \mathrm{Im}(q))\mathbf{x} \\ \mathbf{x}^H\mathbf{x} - 1 \end{bmatrix}, \qquad \mathbf{v} = \begin{bmatrix} q \\ \mathbf{x} \end{bmatrix}. \tag{6.5.4}$$

which is fully constrained. Whether this reformulation is numerically effective remains to be seen. However, the approach of reformulating the real two-parameter eigenproblem into a complex one-parameter problem is very interesting, and is something we will discuss in Chapter 7.

This application of Newton's method to the multiparameter eigenvalue problems has been presented previously in [4], though this reference modifies the algorithm further to become the tensor Rayleigh quotient iteration, and does not consider the case $\mathrm{A}_2 = \overline{\mathrm{A}}_1$. However, in an unusual and apparently unmotivated variant, Khazanov [10] applies this method to a system with one linear q-parameter eigenvalue problem of size $n \times n$ coupled with a constraint equation of the form $\mathrm{C}(\mathbf{x})\mathbf{x} = 0$, where $\mathbf{x}$ is the eigenvalue of the linear problem and C is of size $q \times n$, $q < n$. However, this implementation on Newton's method is new to aeroelasticity.

6.5.3 Tensor Rayleigh quotient iteration (TRQI)

The tensor Rayleigh quotient iteration is an iterative method for linear two-parameter eigenvalue problems that is based around modifying the full eigenvalue / eigenvector iteration to use the information from the Rayleigh quotient of the solution at the previous timestep. For an eigenvalue problem $\mathrm{A}\mathbf{x} = \lambda\mathrm{B}\mathbf{x}$, the Rayleigh quotient provides a further relationship between the eigenvalues and eigenvectors:

$$\lambda = \rho(\mathbf{x}) = \frac{\mathbf{x}^H\mathrm{A}\mathbf{x}}{\mathbf{x}^H\mathrm{B}\mathbf{x}}. \tag{6.5.5}$$

We will not go over the derivation of the TRQI algorithm, but essentially it reduces the full eigenvalue / eigenvector Newton iteration to an iteration that acts only on the eigenvalues (making for a much smaller system). Plestenjak [4] devises the TRQI algorithm but uses alongside a continuation method to solve a linear problem. The method is extendable to

multiparameter problems, though the algebraic expressions involved become increasingly unwieldy. We note that the singular vector iteration that we present in Section 6.6 borrows many elements of tensor Rayleigh quotient iteration, but also has significant differences (representing a Picard-type implementation instead of a Newton-type).

6.6 SINGULAR VECTOR ITERATION (SVI)

6.6.1 Initial manoeuvres

Now we consider a quite different algorithm, which as far as we know does not correspond to any existing one-parameter or multiparameter algorithm, but which does have a few qualities which recommend it for computing flutter points. Consider the nonlinear eigenvalue problem

$$\begin{aligned} \mathrm{A}(\chi, p)\mathbf{x} &= \mathbf{0} \\ \overline{\mathrm{A}}(\chi, p)\bar{\mathbf{x}} &= \mathbf{0}, \end{aligned} \tag{6.6.1}$$

where we assume $\mathrm{A}(\chi, p)$ is continuously differentiable. If we were to know an exact eigenvalue $[\chi^*, p^*]$, then it would follow that the corresponding eigenvector $\mathbf{x}^*$ is given by the nullspace of $\mathrm{A}(\chi^*, p^*)$, because of course $\mathrm{A}(\chi^*, p^*)\mathbf{x}^* = 0$. In general we will not know any eigenvalues exactly. However, we may have an estimate to an eigenvalue, $[\chi_0, p_0]$. We might then be interested in computing $\mathbf{x}_0$ such that $\mathrm{A}(\chi_0, p_0)\mathbf{x}_0 = 0$, as if $[\chi_0, p_0]$ is close to $[\chi^*, p^*]$ then we would expect $\mathbf{x}_0$ to be close to $\mathbf{x}^*$. If Eq. 6.6.1 is nonsingular (as is generally the case) then no such $\mathbf{x}_0 \neq 0$ exists, because $\mathrm{A}(\chi_0, p_0)$ will still have full rank and thus zero nullity (by the Rank–nullity theorem [15]). However, we can estimate $\mathbf{x}_0$ such that $\mathrm{A}(\chi_0, p_0)\mathbf{x}_0 \approx 0$ using the singular value decomposition. We will first introduce the singular value decomposition.

6.6.2 The singular value decomposition

The singular value decomposition (SVD) of a matrix $\mathrm{A} \in \mathbb{C}^{m\times n}$ is the decomposition

$$\mathrm{M} = \mathrm{U}\Sigma\mathrm{V}^H, \tag{6.6.2}$$

where U is an $m \times m$ unitary matrix, V is an $n \times n$ unitary matrix, and Σ is an $m \times n$ rectangular diagonal matrix. In this section we will only be concerned with the case $m = n$, where all matrices are square. The columns of U and V are, respectively, the left and right

singular vectors of M, and the diagonal entries of Σ are the corresponding singular values (σ_i). The SVD can be computed using in-built functions on many computing platforms, including MATLAB, and so we will not go over the methods used to compute it. What is of relevance to us is the relationship between the SVD and our multiparameter eigenvalue problem.

In the case where M is singular – for example $\mathrm{M} = \mathrm{A}(\chi^*, p^*)$ where $[\chi^*, p^*]$ is the exact eigenvalue we were discussing in Section 6.6.1 – then at least one of the singular values will be zero. The number of singular values that are zero is equivalent to the nullity of M, and the right singular vectors corresponding to these singular values form the (right) nullspace of M – and thus the eigenvector(s) corresponding to $[\chi^*, p^*]$. However, unlike the nullspace, the singular value decomposition can be evaluated for nonsingular matrices – such as $\mathrm{M} = \mathrm{A}(\chi_0, p_0)$. In this case none of the singular values will be exactly zero, but one or more of them may be near zero. The corresponding right singular vectors are then estimates of eigenvectors corresponding to $[\chi^*, p^*]$.

6.6.3 The Rayleigh quotient

As may be noticed, we now have the beginnings of an iterative scheme. If we start with an initial estimate of the eigenvalue, $[\chi_k, p_k]$, we can, via the process we have just outlined, compute an estimate of the eigenvector, $\mathbf{x}_k$. However, we would then like to use this eigenvector estimate to update our original eigenvalue estimate. A good candidate for this procedure is the computation of the Rayleigh quotient, which in linear one-parameter eigenvalue problems gives an estimate of the eigenvalue corresponding to a given eigenvector. As noted earlier, for a generalised eigenvalue problem $\mathrm{A}\mathbf{x} = \lambda\mathrm{B}\mathbf{x}$, we have

$$\lambda = \frac{\mathbf{x}^H \mathrm{A}\mathbf{x}}{\mathbf{x}^H \mathrm{B}\mathbf{x}} \tag{6.6.3}$$

which is the Rayleigh quotient. Eq. 6.6.3 follows from multiplying each side of the generalised eigenvalue equation by $\mathbf{x}^H$. However, in the nonlinear multiparameter case it is difficult or impossible to define an analogue to Eq. 6.6.3. We therefore linearise our matrix function using a first-order multivariate Taylor series about $[\chi_k, p_k]$, [6]:

$$\mathrm{A}(\chi,p)=\mathrm{A}(\chi_k,p_k)+(\chi-\chi_k)\partial_\chi\mathrm{A}(\chi_k,p_k)+(p-p_k)\partial_p\mathrm{A}(\chi_k,p_k), \tag{6.6.4}$$

transforming our original nonlinear multiparameter eigenvalue problem into

$$\begin{aligned}(\mathrm{A}_k+\Delta\chi_k\mathrm{X}_k+\Delta p_k\mathrm{P}_k)\mathbf{x}=0\\(\overline{\mathrm{A}}_k+\Delta\chi_k\overline{\mathrm{X}}_k+\Delta p_k\overline{\mathrm{P}}_k)\bar{\mathbf{x}}=0,\end{aligned} \tag{6.6.5}$$

with

$$\begin{aligned}\mathrm{A}_k&=\mathrm{A}(\chi_k,p_k)\\\mathrm{X}_k&=\partial_\chi\mathrm{A}(\chi_k,p_k)\\\mathrm{P}_k&=\partial_p\mathrm{A}(\chi_k,p_k),\end{aligned} \tag{6.6.6}$$

and eigenvalues

$$\begin{aligned}\Delta\chi_k&=\chi-\chi_k\\\Delta p_k&=p-p_k.\end{aligned} \tag{6.6.7}$$

This is the same linearisation we performed when devising the SLP algorithm. The matrix derivatives may be computed using finite differences if nothing further is known about $\mathrm{A}(\chi,p)$. However, here we are not interested in solving this linearised problem for $[\Delta\chi_k,\Delta p_k]$, but in using the approximate eigenvector computed in the singular value decomposition ($\mathbf{x}_k$) to estimate a corresponding $[\Delta\chi_k,\Delta p_k]$. To do this we transform the system to a generalised eigenvalue problem via the operator determinants introduced in Chapter 4;

$$\begin{aligned}\Delta_{0,k}&=\mathrm{X}_k\otimes\overline{\mathrm{P}}_k-\mathrm{P}_k\otimes\overline{\mathrm{X}}_k\\\Delta_{1,k}&=\mathrm{P}_k\otimes\overline{\mathrm{A}}_k-\mathrm{A}_k\otimes\overline{\mathrm{P}}_k\\\Delta_{2,k}&=\mathrm{A}_k\otimes\overline{\mathrm{X}}_k-\mathrm{X}_k\otimes\overline{\mathrm{A}}_k.\end{aligned} \tag{6.6.8}$$

in which case

$$\begin{aligned}\Delta_{1,k}\mathbf{z}&=\Delta\chi_k\Delta_{0,k}\mathbf{z}\\\Delta_{2,k}\mathbf{z}&=\Delta p_k\Delta_{0,k}\mathbf{z},\end{aligned} \tag{6.6.9}$$

with $\mathbf{z}=\mathbf{x}\otimes\bar{\mathbf{x}}$. These are generalised eigenvalue problems, and as noted earlier we can define Rayleigh quotients for them

$$\Delta\chi_k = \frac{\mathbf{z}^H \Delta_{1,k} \mathbf{z}}{\mathbf{z}^H \Delta_{0,k} \mathbf{z}}$$
$$\Delta p_k = \frac{\mathbf{z}^H \Delta_{2,k} \mathbf{z}}{\mathbf{z}^H \Delta_{0,k} \mathbf{z}}. \tag{6.6.10}$$

We can then take $\mathbf{z}$ as $\mathbf{z}_k = \mathbf{x}_k \otimes \bar{\mathbf{x}}_k$, and update our estimate of $[\chi_k, p_k]$ by

$$\chi_{k+1} = \Delta\chi_k + \chi_k$$
$$p_{k+1} = \Delta p_k + p_k \tag{6.6.11}$$

We thus have a sketch of an iterative algorithm. The convergence this algorithm to $[\chi^*, p^*]$ is not guaranteed at this stage, but we will analyse it numerically in Section 6.6.5. Algorithm 6.5 presents such an algorithm, using finite differences to compute P_k and X_k, and using a convergence criterion based on both the Euclidean increment $\Delta\chi_k^2 + \Delta p_k^2$ and the two-norm of the residual at k, $r_k = \|\mathrm{A}(\chi_k, p_k)\mathbf{x}_k\|_2$. Some commentary is included due to the novel nature of the algorithm.

Algorithm 6.5 – sketch of an SVI algorithm

1	**initialise** $k = 0$, initial guesses χ_0, p_0, tolerances ϵ_1, ϵ_2	
2	evaluate $\mathrm{A}_k = \mathrm{A}(\chi_k, p_k)$	
3	compute U_k, Σ_k, V_k such that $\mathrm{A}_k = \mathrm{U}_k \Sigma_k \mathrm{V}_k^H$	Compute the singular value decomposition (Section 6.6.2).
4	select a column of V_k and denote it $\mathbf{x}_k$	Discussed in Section 6.6.4.
5	evaluate $\mathrm{P}_k = \frac{\mathrm{A}(\chi_k, p_k + \delta p) - \mathrm{A}_k}{\delta p}$ $\mathrm{X}_k = \frac{\mathrm{A}(\chi_k + \delta\chi, p_k) - \mathrm{A}_k}{\delta\chi}$	Compute matrix derivatives via finite differences.
6	evaluate $\Delta_{0,k} = \mathrm{X}_k \otimes \overline{\mathrm{P}}_k - \mathrm{P}_k \otimes \overline{\mathrm{X}}_k$ $\Delta_{1,k} = \mathrm{P}_k \otimes \overline{\mathrm{A}}_k - \mathrm{A}_k \otimes \overline{\mathrm{P}}_k$ $\Delta_{2,k} = \mathrm{A}_k \otimes \overline{\mathrm{X}}_k - \mathrm{X}_k \otimes \overline{\mathrm{A}}_k.$ $\mathbf{z}_k = \mathbf{x}_k \otimes \bar{\mathbf{x}}_k$	Compute the operator determinants of the linearised system.

7	evaluate $\Delta\chi_k = \dfrac{\mathbf{z}^H \Delta_{1,k} \mathbf{z}}{\mathbf{z}^H \Delta_{0,k} \mathbf{z}}$, $\Delta p_k = \dfrac{\mathbf{z}^H \Delta_{2,k} \mathbf{z}}{\mathbf{z}^H \Delta_{0,k} \mathbf{z}}$	Compute the Rayleigh quotients, giving us the increment to update the iterate.
8	evaluate $\chi_{k+1} = \Delta\chi_k + \chi_k$, $p_{k+1} = \Delta p_k + p_k$	Update iterate and test convergence.
9	**if** $\Delta\chi_k^2 + \Delta p_k^2 < \epsilon_1$ and $\|\mathrm{A}(\chi_k, p_k)\mathbf{x}_k\|_2 < \epsilon_2$	
10	**then** return p_k, χ_k and $\mathbf{x}_k$	
11	**else** $k = k + 1$, **goto** (2)	

6.6.4 Implementation and variants

The most interesting thing about this algorithm is that it converges at all (as we will show in the numerical experiments). There appears to be no reason to believe that the new estimate of the flutter point provided by the Rayleigh quotient will be any more accurate than the estimate that is used to initiate each iteration – one is simply computing the Rayleigh quotient corresponding (approximately) to an existing estimate of the eigenvector. A theoretical convergence analysis is beyond the scope of this thesis, but we will analyse its convergence numerically in Section 6.6.5. However, before we do this, there are several things in Algorithm 6.5 that need explanation or can be improved.

Firstly, the Rayleigh quotient in line 7 can, in the two-parameter case, be transformed into a more efficient form which does not involve working with the large operator determinants. Substituting the definitions of the two-parameter operator determinants and $\mathbf{z}$ into the Rayleigh quotient (Eq. 6.6.3) we obtain

$$\begin{aligned} \Delta\chi_k &= \frac{(\mathbf{x}_k \otimes \bar{\mathbf{x}}_k)^H (\mathrm{P}_k \otimes \overline{\mathrm{A}}_k - \mathrm{A}_k \otimes \overline{\mathrm{P}}_k)(\mathbf{x}_k \otimes \bar{\mathbf{x}}_k)}{(\mathbf{x}_k \otimes \bar{\mathbf{x}}_k)^H (\mathrm{X}_k \otimes \overline{\mathrm{P}}_k - \mathrm{P}_k \otimes \overline{\mathrm{X}}_k)(\mathbf{x}_k \otimes \bar{\mathbf{x}}_k)} \\ \Delta p_k &= \frac{(\mathbf{x}_k \otimes \bar{\mathbf{x}}_k)^H (\mathrm{A}_k \otimes \overline{\mathrm{X}}_k - \mathrm{X}_k \otimes \overline{\mathrm{A}}_k)(\mathbf{x}_k \otimes \bar{\mathbf{x}}_k)}{(\mathbf{x}_k \otimes \bar{\mathbf{x}}_k)^H (\mathrm{X}_k \otimes \overline{\mathrm{P}}_k - \mathrm{P}_k \otimes \overline{\mathrm{X}}_k)(\mathbf{x}_k \otimes \bar{\mathbf{x}}_k)}. \end{aligned} \tag{6.6.12}$$

This can be simplified using the mixed-product property, namely that

$$(\mathrm{A} \otimes \mathrm{B}) \times (\mathrm{C} \otimes \mathrm{D}) = (\mathrm{A} \times \mathrm{C}) \otimes (\mathrm{B} \times \mathrm{D}), \tag{6.6.13}$$

which, given that $(\mathrm{A} \otimes \mathrm{B})^H = \mathrm{A}^H \otimes \mathrm{B}^H$, implies

$$\begin{aligned} \Delta\chi_k &= \frac{\mathbf{x}_k^H \mathrm{P}_k \mathbf{x}_k \otimes \bar{\mathbf{x}}_k^H \overline{\mathrm{A}}_k \bar{\mathbf{x}}_k - \mathbf{x}_k^H \mathrm{A}_k \mathbf{x}_k \otimes \bar{\mathbf{x}}_k^H \overline{\mathrm{P}}_k \bar{\mathbf{x}}_k}{\mathbf{x}_k^H \mathrm{X}_k \mathbf{x}_k \otimes \bar{\mathbf{x}}_k^H \overline{\mathrm{P}}_k \bar{\mathbf{x}}_k - \mathbf{x}_k^H \mathrm{P}_k \mathbf{x}_k \otimes \bar{\mathbf{x}}_k^H \overline{\mathrm{X}}_k \bar{\mathbf{x}}_k} \\ \Delta p_k &= \frac{\mathbf{x}_k^H \mathrm{A}_k \mathbf{x}_k \otimes \bar{\mathbf{x}}_k^H \overline{\mathrm{X}}_k \bar{\mathbf{x}}_k - \mathbf{x}_k^H \mathrm{X}_k \mathbf{x}_k \otimes \bar{\mathbf{x}}_k^H \overline{\mathrm{A}}_k \bar{\mathbf{x}}_k}{\mathbf{x}_k^H \mathrm{X}_k \mathbf{x}_k \otimes \bar{\mathbf{x}}_k^H \overline{\mathrm{P}}_k \bar{\mathbf{x}}_k - \mathbf{x}_k^H \mathrm{P}_k \mathbf{x}_k \otimes \bar{\mathbf{x}}_k^H \overline{\mathrm{X}}_k \bar{\mathbf{x}}_k} \end{aligned} \tag{6.6.14}$$

and so

$$\begin{aligned} \Delta\chi_k &= \frac{(\mathbf{x}_k^H \mathrm{P}_k \mathbf{x}_k)(\bar{\mathbf{x}}_k^H \overline{\mathrm{A}}_k \bar{\mathbf{x}}_k) - (\mathbf{x}_k^H \mathrm{A}_k \mathbf{x}_k)(\bar{\mathbf{x}}_k^H \overline{\mathrm{P}}_k \bar{\mathbf{x}}_k)}{(\mathbf{x}_k^H \mathrm{X}_k \mathbf{x}_k)(\bar{\mathbf{x}}_k^H \overline{\mathrm{P}}_k \bar{\mathbf{x}}_k) - (\mathbf{x}_k^H \mathrm{P}_k \mathbf{x}_k)(\bar{\mathbf{x}}_k^H \overline{\mathrm{X}}_k \bar{\mathbf{x}}_k)} \\ \Delta p_k &= \frac{(\mathbf{x}_k^H \mathrm{A}_k \mathbf{x}_k)(\bar{\mathbf{x}}_k^H \overline{\mathrm{X}}_k \bar{\mathbf{x}}_k) - (\mathbf{x}_k^H \mathrm{X}_k \mathbf{x}_k)(\bar{\mathbf{x}}_k^H \overline{\mathrm{A}}_k \bar{\mathbf{x}}_k)}{(\mathbf{x}_k^H \mathrm{X}_k \mathbf{x}_k)(\bar{\mathbf{x}}_k^H \overline{\mathrm{P}}_k \bar{\mathbf{x}}_k) - (\mathbf{x}_k^H \mathrm{P}_k \mathbf{x}_k)(\bar{\mathbf{x}}_k^H \overline{\mathrm{X}}_k \bar{\mathbf{x}}_k)} \end{aligned} \tag{6.6.15}$$

because $\mathbf{x}_k^H \mathrm{A}_k \mathbf{x}_k$ etc. are scalar. When the matrices involved are large, Eq. 6.6.15 is significantly easier to evaluate than Eq. 6.6.10, which requires the evaluation of Kronecker products of these matrices, whereas Eq. 6.6.15 requires only matrix multiplication. Note that the numerators and denominators of Eq. 6.6.15 are in the form of determinants. In the case of the two-parameter problems we are dealing with there is no real motivation to set up such a determinantal representation; however, such a representation would be very valuable when working with an n-parameter system.

There are two other aspects of Algorithm 6.5 that should be discussed: the selection of a suitable $\mathbf{x}_k$ from V_k (line 4), and the computation of V_k (line 3). As to the selection, there is really only one sensible choice: to select the singular vector corresponding to the minimum singular value, as this singular value is the closest to zero and thus the closest to a flutter point. This corresponds to the rightmost column of V_k, if the standard ordering for U_k, Σ_k and V_k is used (Σ_k ordered from largest to smallest along the diagonal). If this selection criterion is used, then a simple iterative algorithm can be used to compute $\mathbf{x}_k$. Given that the singular vectors of a matrix A correspond to the eigenvectors of the matrix $(\mathrm{A}^H \mathrm{A})^{1/2}$ [11,15], we can compute $\mathbf{x}_k$ via inverse iteration [7]. The sequence

$$\mathbf{y}_{i+1} = \frac{(\mathrm{A}_k^H \mathrm{A}_k)^{-1/2} \mathbf{y}_i}{\|(\mathrm{A}_k^H \mathrm{A}_k)^{-1/2} \mathbf{y}_i\|} \tag{6.6.16}$$

for some norm $\|\cdot\|$ will generally converge to our desired $\mathbf{x}_k$, from a random $\mathbf{y}_0$. We use zero shift because, of course, we are only interested in converging to the singular vector associated with minimum singular value (i.e. the singular value closest to zero). Inverse iteration is noted to have remarkably resilient convergence [7] and so the selection of $\mathbf{y}_0$ is not of great importance.

6.6.5 Numerical experiments

As per usual, we simulate the nondimensional section model with Theodorsen aerodynamics, in χ-Υ form. Parameter values are taken from Chapter 2. Figure 6.18 shows a contour plot with nine different iteration paths (from nine different initial guesses) superimposed. Five of the iteration paths converge to the first flutter point, one to flutter points with negative airspeed values, and three do not converge within the maximum number of iterations (40). The tolerance is $\epsilon_1 = \epsilon_2 = 10^{-4}$. However, it is easy to see that at least one of these supposedly nonconvergent iterations is actually converging (to the divergence point, albeit slowly). Figure 6.19 shows the same contour pot and iteration paths, but with the maximum number of iterations before algorithm termination set to 200. As can be seen, the previously nonconvergent iterations are now recorded as having converged – one to the divergence point, and two to the first flutter point. Figure 6.20 shows a logarithmic convergence plot for an iteration converging to the first flutter point. The convergence rate is uniform, but only first-order (plot gradient 0.985). Each step of the singular vector iteration only decreases the residual by a small amount. However, it should be noted that the application of a single step of the singular vector iteration does not require much computing time – only about 2 milliseconds per iterate in a standard MATLAB workspace. Figure 6.21 shows a logarithmic convergence plot for an iteration converging to the divergence point. Again the convergence rate is uniform, first-order (gradient 1.08) and slow. The contrast between the singular vector iteration's convergence to the divergence point and the convergence of the other iterative methods that we have discussing is remarkable – compare Figure 6.21 to Figures 6.3, 6.11 and 6.16. The SVI algorithm's convergence is monotonic, but much slower in terms of the number of iterations required to reach a given convergence criterion.

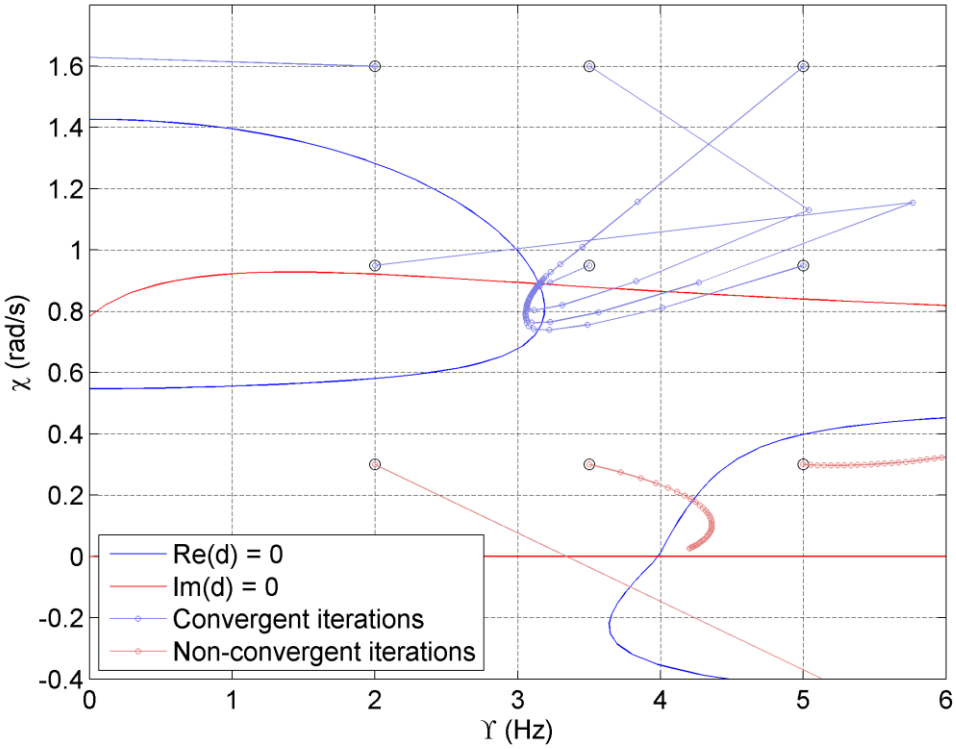


Figure 6.18: Nine iteration paths of the SVI algorithm (Υ-χ section model, terminated after 40 iterations).

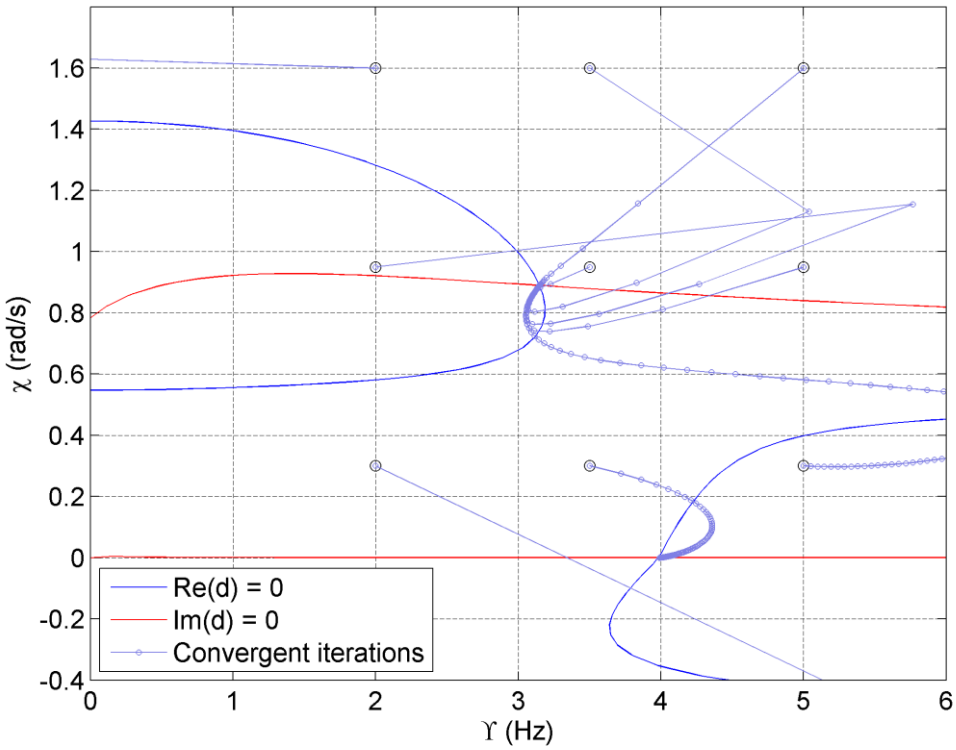


Figure 6.19: Nine iteration paths of the SVI algorithm (Υ-χ section model, terminated after 200 iterations).

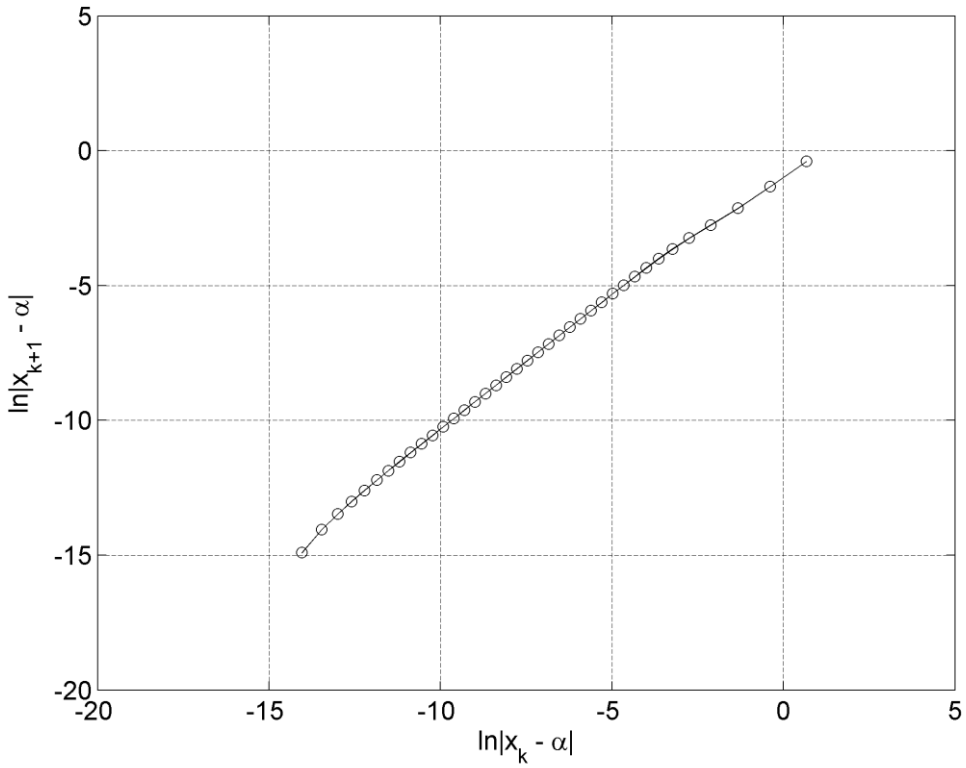


Figure 6.20: Logarithmic convergence plot for the singular vector iteration converging to the flutter point ($\chi_0 \ = \ 1.5 \ \mathrm{rad/s}, \Upsilon_0 \ = \ 5 \ \mathrm{Hz}$).

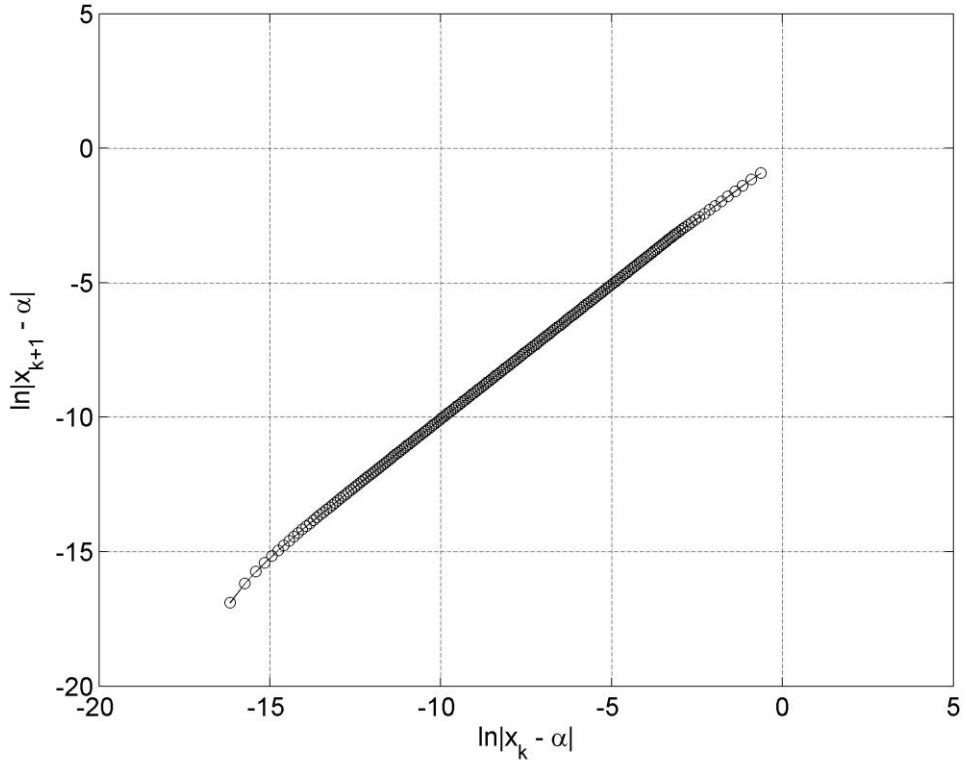


Figure 6.21: Logarithmic convergence plot for the singular vector iteration converging to the divergence point ($\chi_0 \ = \ 0.2 \ \mathrm{rad/s}, \Upsilon_0 \ = \ 3.5 \ \mathrm{Hz}$).

Figures 6.22, 6.23 and 6.24 show numerical convergence analyses of the singular vector iteration (SVI) algorithm, with an increasing maximum-iterations limit. As can be seen, the basin of attraction to the first flutter point is very extensive, though convergence is very slow in some locations. Nevertheless, even the domain of points that converge within 40 iterations (Figure 6.22) is significant. Figure 6.25 shows a numerical convergence analysis with a wide field of view. As can be seen, the basin of attraction of the first flutter point dominates the upper-right quarter plane. In terms of its convergence to the first flutter point, this method is the best by a significant margin – compare these numerical convergence analyses with those of Section 6.2.3, Section 6.3.2 and Section 6.4.3. Note that we are still working with the Υ-χ form and have not even needed to shift to the τ-λ form to get good first flutter point convergence.

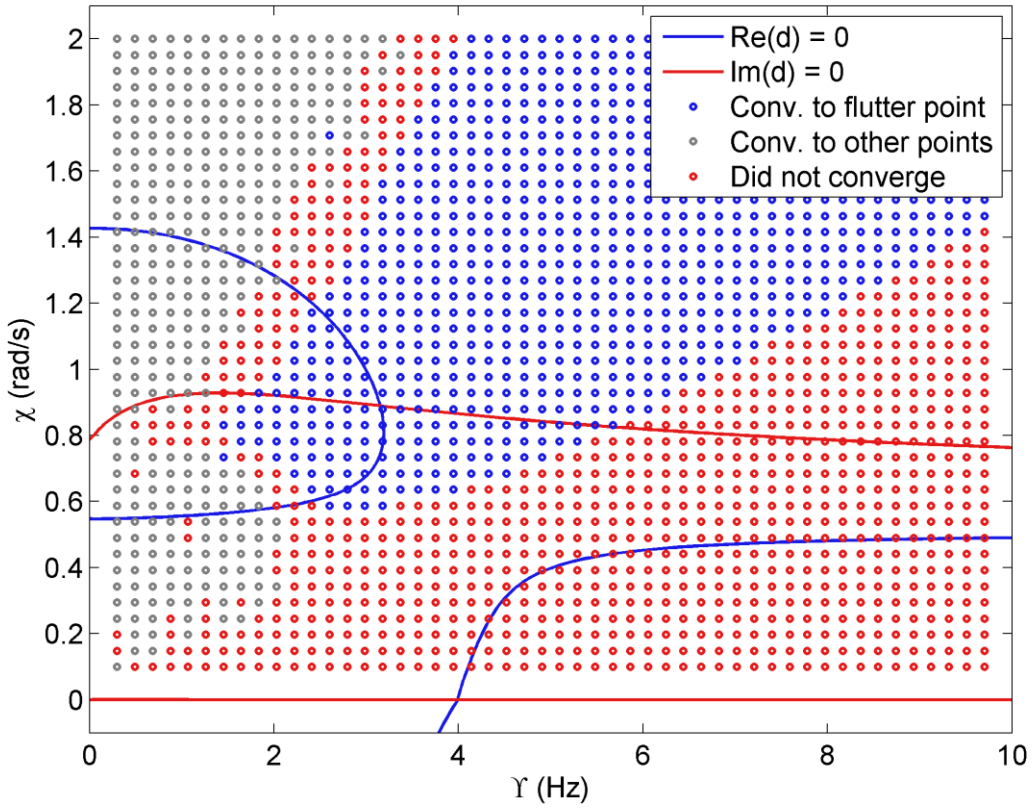


Figure 6.22: Numerical convergence analysis of the singular vector iteration (Υ-χ section model, maximum iterations: 40).

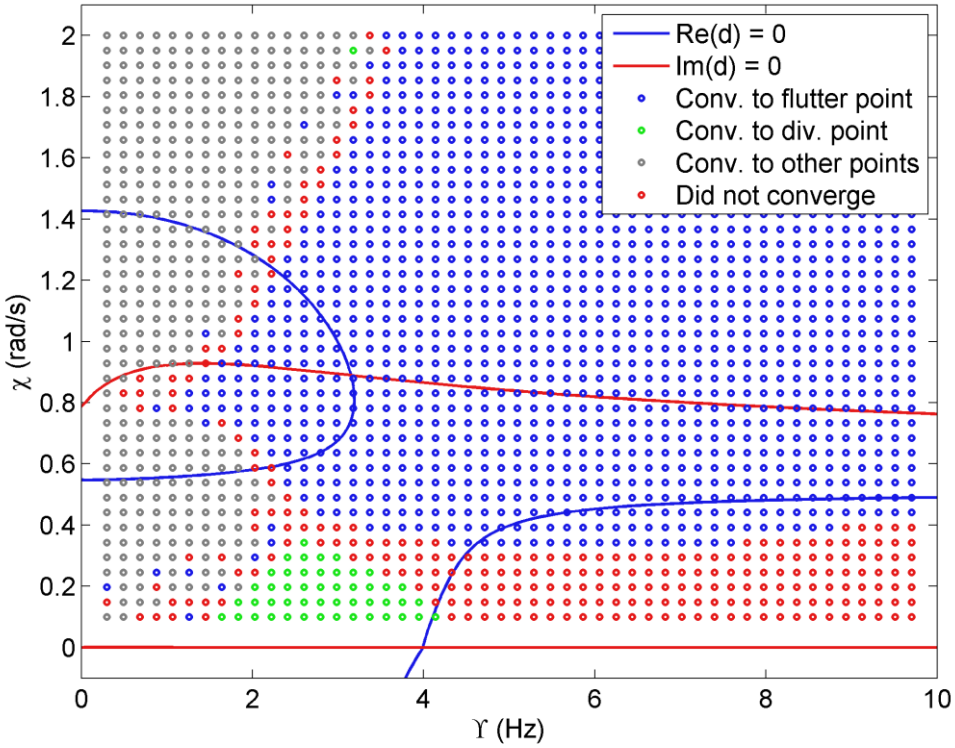


Figure 6.23: Numerical convergence analysis of the singular vector iteration (Υ-χ section model, maximum iterations: 100).

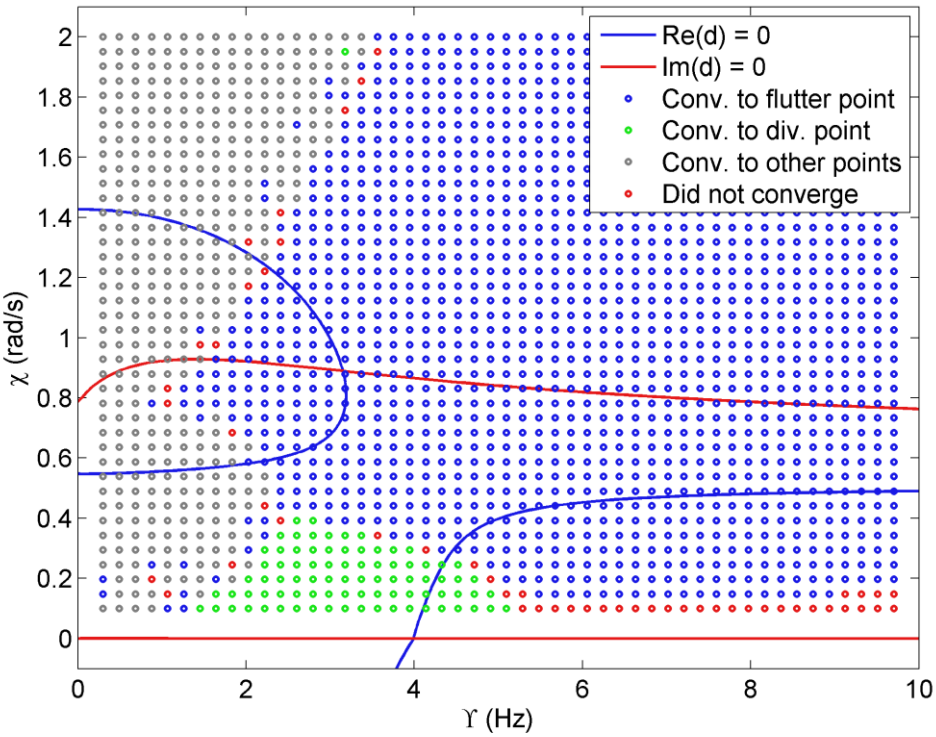


Figure 6.24: Numerical convergence analysis of the singular vector iteration (Υ-χ section model, maximum iterations: 500).

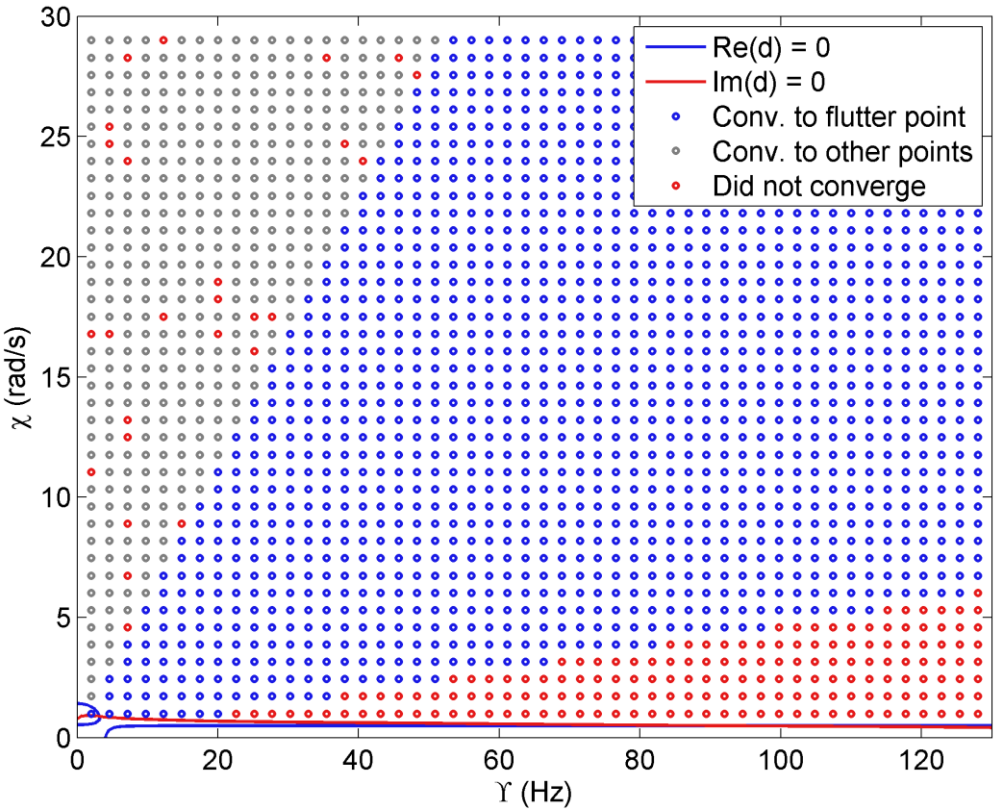


Figure 6.25: Numerical convergence analysis of the singular vector iteration (Υ-χ section model, maximum iterations: 100, wide field of view).

6.7 RESTRICTION

In this chapter we have concentrated (especially in our numerical convergence analyses) on computing physical flutter points in the upper-right quarter plane. However, as we showed in Chapter 3, the section model with Theodorsen aerodynamics has many more flutter points outside this quarter plane. We are not interested in these flutter points: they are not physical and serve only to distract our iterative methods. The behaviour of this model is actually very messy outside the upper-right quarter plane – see Chapter 3 – and there are two reasons why we might want to try to eliminate this behaviour.

Firstly, when we come to the study of deflation methods for our algorithms (Section 6.8) we will see that these deflation methods become increasingly problematic as more flutter points are deflated. It is therefore prudent if we can minimise the number of flutter points outside the upper-right quarter plane. Secondly, and more subtly, we might note that, in all our methods, quite a number of iterations pass through the lower half plane ($\chi < 0$) and

then converge to a physical flutter point (especially the divergence point). If the nonlinear matrix function $\mathrm{A}(\chi, p)$ involved some form of computational process (e.g. computational fluid dynamics) how would we perform this computational process with negative χ? In many cases this would not be possible. We could simply terminate the iteration when it reached the lower half plane, but this means losing some iterates which could have converged.

There is, however, another possibility, and that is to define the value of the matrix function outside the upper-right quarter plane (or any other domain of interest) based on extrapolation from information inside that domain. The most obvious extrapolation formula available to us is the Taylor expansion. We devise, therefore, the following modification to the matrix function $\mathrm{A}(\chi, p)$, which henceforth we term a *restriction* method:

$$\mathrm{A}^*(\chi,p)=\begin{cases}\mathrm{A}(\chi,p) & \chi\geq\chi_c, p\geq p_c\\ \mathrm{A}(\chi_c,p)+(\chi-\chi_c)\left.\dfrac{\partial \mathrm{A}}{\partial\chi}\right|_{(\chi_c,p)} & \chi<\chi_c, p\geq p_c\\ \mathrm{A}(\chi,p_c)+(p-p_c)\left.\dfrac{\partial \mathrm{A}}{\partial p}\right|_{(\chi,p_c)} & \chi\geq\chi_c, p<p_c\\ \mathrm{A}(\chi_c,p_c)+(p-p_c)\left.\dfrac{\partial \mathrm{A}}{\partial p}\right|_{(\chi_c,p_c)}+(\chi-\chi_c)\left.\dfrac{\partial \mathrm{A}}{\partial\chi}\right|_{(\chi_c,p_c)} & \chi<\chi_c, p<p_c\end{cases} \tag{6.7.1}$$

where χ_c and p_c define the boundary between the domain that is unmodified and the modified domain. One sensible choice for this boundary is $\chi_c = p_c = 0$, as this simplifies the areas where the flutter points are nonphysical (negative airspeed and frequency) but does not infringe on any areas where the flutter points are still physical (positive airspeed and frequency). This also eliminates the problem that we noted earlier of sometimes having to compute the system matrix for these negative parameter values. However, this is not the only choice, as if we were interested in a particular high-airspeed or high-frequency flutter point, we could choose some $\chi_c > 0$, $p_c > 0$ so as to simplify the system's behaviour at frequencies and airspeeds below this boundary. This would be particularly useful for computing the flutter points of continuous systems (with an infinite or large number of flutter points) as the desired flutter point can be singled out.

Note that the restriction method will necessarily make any linear system into a nonlinear one, and so cannot be applied to the direct solvers of Chapter 4. The best way of explaining how Eq. 6.7.1 functions is to use a diagram – see Figure 6.26. Note that for compactness we have show the case $\chi_c < 0, p_c < 0$, which will not often be used in practice.

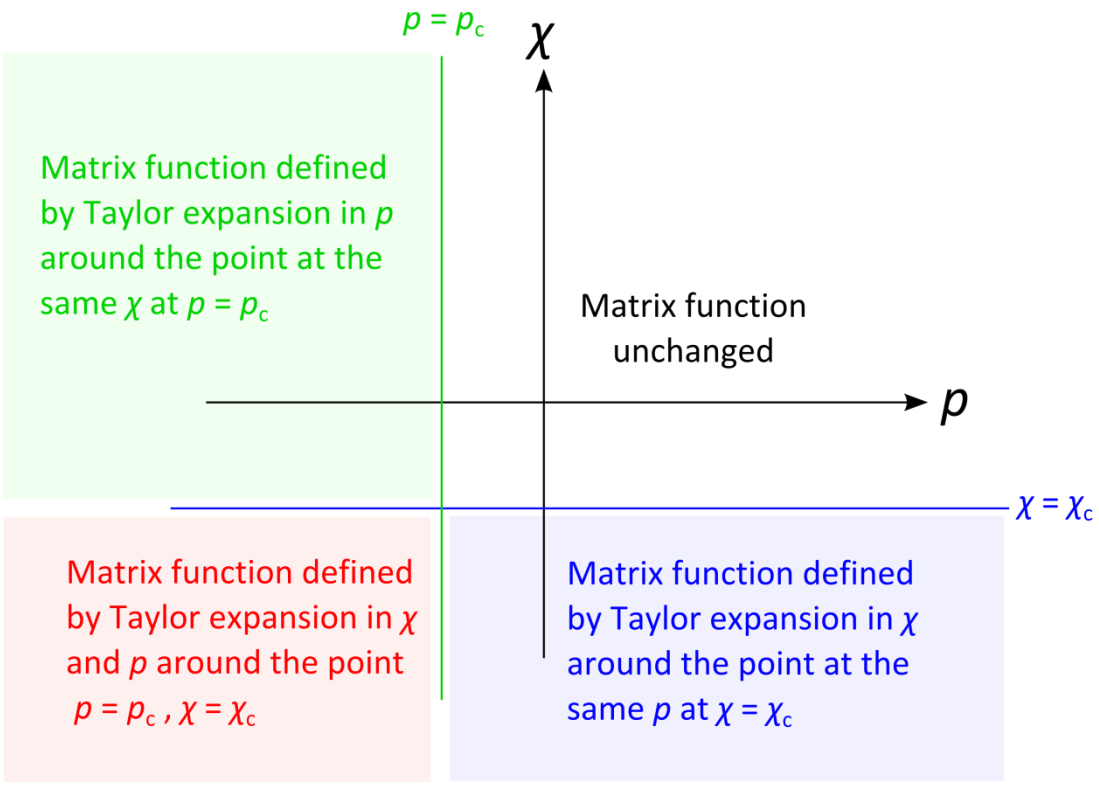


Figure 6.26: Diagram of restriction process

We will term the area where the matrix function remains unchanged as the *unrestricted zone*, and all other areas as the *restricted zone*. The Taylor expansion used to approximate the restricted zone is continuous and continuously differentiable, meaning that it is suitable for use with methods that perform first-order Taylor expansions of the matrix function $\mathrm{A}(\chi, p)$. It is not certain whether it is suitable for methods which use higher-order Taylor expansions of the matrix function: it may cause the method to fail to converge, or only to converge at a slower rate that it would otherwise. As we noted before, the choice of χ_c and p_c is arbitrary: when they are both zero then only the upper-right quarter plane is unrestricted; otherwise the field of restriction may be extended or diminished. It is easy to derive variants of the method that restrict an upper bound on each axes (isolating a rectangle of finite volume in p-χ space) or even a space of different shape (triangles, etc.). It

is also easy to generalise this restriction method to higher-dimension problems (the Taylor series simply acquires more terms). However, for our purposes, Eq. 6.7.1 will be sufficient.

Eq. 6.7.1 does require the evaluation of the derivatives: this is easy to perform with finite-differences. In terms of what type of difference we chose: it makes sense to ensure that all points used to compute the difference lie inside the unrestricted zone or exactly on its boundary, as this will prevent information from the restricted zone to propagate back into the system. This allows the method to be applied right on the line of a discontinuity (e.g. the divergence point discontinuity we noted in Chapter 3), thus eliminating the discontinuity. First-order forward differences are suitable:

$$\frac{\partial \mathrm{A}}{\partial \chi} = \frac{A(\chi + \Delta\chi, p) - A(\chi, p)}{\Delta\chi}$$
$$\frac{\partial \mathrm{A}}{\partial p} = \frac{A(\chi, p + \Delta p) - A(\chi, p)}{\Delta p} \qquad (6.7.2)$$

These derivatives are always evaluated either along $\chi = \chi_c$ or $p = p_c$. Considering our section model with Theodorsen aerodynamics, if we set $\chi_c = p_c = 0.1$ ($p \equiv \Upsilon$) we can eliminate both the discontinuity at the divergence point and the oscillations at negative Υ. We do not choose $\chi_c = \Upsilon_c = 0$ as evaluating Theodorsen's function exactly at this value is not numerically possible ($0/0$ being indeterminate). Figure 6.27 shows this system without restriction, and Figure 6.28 shows the system with this proposed restriction. As can be seen, the restricted system is significantly simpler and has fewer nonphysical flutter points. No such restriction algorithm has been presented before, though this is probably because the solution of nonlinear multiparameter eigenvalue problems has not been considered before.

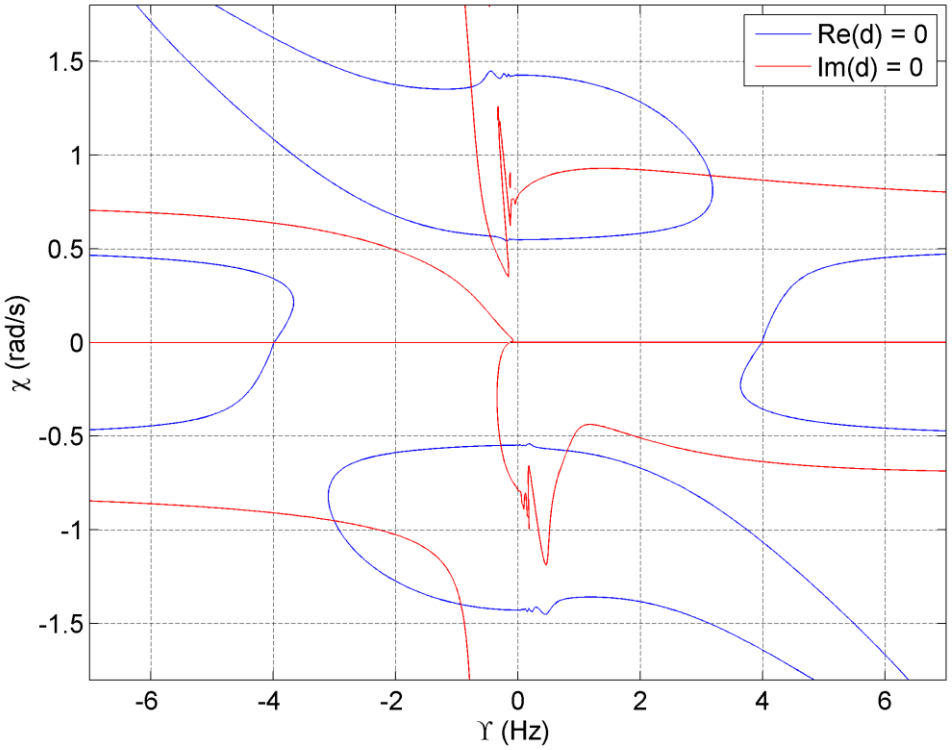


Figure 6.27: Contour plot of section model with Theodorsen aerodynamics (Υ-χ form).

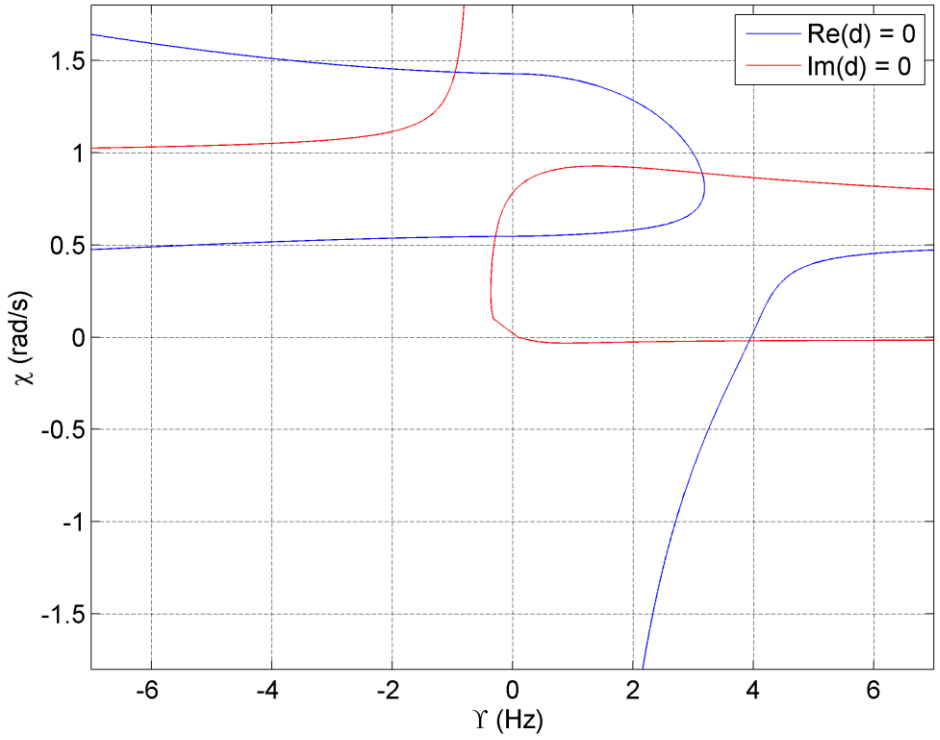


Figure 6.28: Contour plot of section model with Theodorsen aerodynamics (Υ-χ form), restricted with $\chi_c = 0.1$ rad/s and $\Upsilon_c = 0.1$ Hz.

6.8 DEFLATION

6.8.1 Theory

Most aeroelastic systems will have multiple flutter points. We have seen this in Chapter 3. The question then arises of how to compute *all* of these flutter points, or at least a particular set (e.g. those within a certain specified domain, or simply just the first flutter point). When the algorithms involve some iterative component, the iterations can be prevented from converging to the same flutter point twice via the process of deflation. Deflation involves modifying the residual function of the original system such that the root of the residual corresponding to a known flutter point is moved to another location. In the case of our direct solvers in Chapter 4, we need not devote much thought to this process: deflation methods for generalized-eigenproblem solvers are well known and are in place in most numerical-algebra packages [11]. However, given that the iterative algorithms for unstructured multiparameter systems that we have presented in this chapter are new, we must address the question from a more fundamental standpoint.

No deflation techniques for the nonlinear multiparameter eigenvalue problems are present in the literature. In the context of the Jacobi-Davidson method for linear multiparameter eigenvalue problems (an iterative method, see Chapter 4), it is possible to perform deflation via enforcing the orthogonality of the eigenvectors with respect to the operator determinant Δ_0 (See Chapter 4). However, it is not certain whether this approach can be generalised to the nonlinear case, where eigenvector orthogonality is not guaranteed. We thus turn to general nonlinear-system deflation, for which some literature exists. Brown and Gearhart [16] introduced the key concept of the *deflation matrix*. Consider a nonlinear system with state vector $\mathbf{v}$ and vector residual of length n given by

$$\mathbf{F}(\mathbf{v}) = \mathbf{0}. \tag{6.8.1}$$

The deflation system is defined as having residual given by

$$\mathbf{G}(\mathbf{v}) = \mathrm{M}(\mathbf{v}, \mathbf{v}^*)\mathbf{F}(\mathbf{v}) = \mathbf{0} \tag{6.8.2}$$

for some matrix $\mathrm{M}(\mathbf{v}, \mathbf{v}^*)$, this being the deflation matrix. This matrix is chosen so as to eliminate one of the roots of $\mathbf{F}(\mathbf{v})$, while leaving any other roots unmodified. It will generally be a function of the state vector $\mathbf{v}$, and the state vector of the root being

eliminated ($\mathbf{v}^*$). Deflating multiple roots simply involved repeatedly applying M for the different $\mathbf{v}^*$. Several choices of M have been presented in the literature: four common ones are presented in Table 6.1.

Table 6.1: Deflation matrices

Norm deflation [16,17]	$\mathrm{M} = \dfrac{\mathrm{I}}{\|\mathbf{v} - \mathbf{v}^*\|}$	for any given vector norm $\|\cdot\|$
Power norm deflation [17]	$\mathrm{M} = \dfrac{\mathrm{I}}{\|\mathbf{v} - \mathbf{v}^*\|^p}$	for any given vector norm $\|\cdot\|$ and positive integer power $p \in \mathbb{Z}^+$
Inner product deflation [16]	$\mathrm{M}_{ii} = \dfrac{1}{\mathbf{a}_i \cdot (\mathbf{v} - \mathbf{v}^*)}$	for any given set of vectors $\mathbf{a}_i$
Power norm deflation with shift [17]	$\mathrm{M} = \mathrm{I}\left(\dfrac{1}{\|\mathbf{v} - \mathbf{v}^*\|^p} + \alpha\right)$	for any given vector norm $\|\cdot\|$, positive integer power $p \in \mathbb{Z}^+$ and scalar shift α

Refs. [16,17] give a more detailed overview of methods. We then have to apply them to our iterative methods, and the particular definition of the residuals therein. We will be particularly focused on power norm deflation (and its special case, norm deflation) as these methods are widely known to perform reasonable well [16,17]. The iterated contour plot methods (of arbitrary order) are easy to modify: our residual is defined simply as

$$\mathbf{F}(\mathbf{v}) = \begin{bmatrix} \mathrm{Re}\big(z(\mathbf{v})\big) \\ \mathrm{Im}\big(z(\mathbf{v})\big) \end{bmatrix} = \mathbf{0}, \tag{6.8.3}$$

where $z(\mathbf{v}) = \det\big(\mathrm{D}(\mathbf{v})\big)$ for the eigenvalue problem $\mathrm{D}(\mathbf{v})\mathbf{x} = 0$. Our state vector is defined as $\mathbf{v} = [\chi;\ p]$. For a general $\mathrm{M} \in \mathbb{C}^{2\times 2}$, we have

$$\mathbf{F}_m\,(\mathbf{v}) = \mathrm{M}\mathbf{F}(\mathbf{v}) \begin{bmatrix} M_{11}\,\mathrm{Re}\big(z(\mathbf{v})\big) + M_{12}\,\mathrm{Im}\big(z(\mathbf{v})\big) \\ M_{21}\,\mathrm{Re}\big(z(\mathbf{v})\big) + M_{22}\,\mathrm{Im}\big(z(\mathbf{v})\big) \end{bmatrix} = \mathbf{0}, \tag{6.8.4}$$

where M_{ij} are the elements of M. However, we will generally be concerned with the case when M is a diagonal matrix, and more particularly, a scaled identity matrix. For power norm deflation, the modified residual becomes

$$\mathbf{F}_m\,(\mathbf{v}) = \frac{\mathbf{F}(\mathbf{v})}{\|\mathbf{v}-\mathbf{v}^*\|^p} = \begin{bmatrix} \mathrm{Re}\left(\dfrac{z(\mathbf{v})}{\|\mathbf{v}-\mathbf{v}^*\|^p}\right) \\ \mathrm{Im}\left(\dfrac{z(\mathbf{v})}{\|\mathbf{v}-\mathbf{v}^*\|^p}\right) \end{bmatrix} \tag{6.8.5}$$

which, given that $\det(k\mathrm{D}) = k^n \det(\mathrm{D})$ for scalar k and $\mathrm{D} \in \mathbb{C}^{n\times n}$ of size $n \times n$ [18], can be rewritten as

$$\mathbf{F}_m\,(\mathbf{v}) = \begin{bmatrix} \mathrm{Re}(\det(\mathrm{D}_m(\mathbf{x}))) \\ \mathrm{Im}(\det(\mathrm{D}_m(\mathbf{x}))) \end{bmatrix} \tag{6.8.6}$$

with

$$D_m(\mathbf{x}) = \frac{\mathrm{D}(\mathbf{x})}{\|\mathbf{v}-\mathbf{v}^*\|^{np}} \tag{6.8.7}$$

where n is the length of D ($\mathrm{D} \in \mathbb{C}^{n\times n}$). Note that $\|\mathbf{v}-\mathbf{v}^*\|^p \in \mathbb{R}$ for $p \in \mathbb{Z}^+$. Evaluating Eq. 6.8.7 will probably produce numerical underflow or overflow when n is large, but for small n it provides a convenient way of applying the deflation directly to the eigenproblem matrix function $\mathrm{D}(\mathbf{x})$, without any modifications to the actual algorithms.

For the SLP algorithm and the singular vector iteration, the residual of the system may be defined as the eigenproblem residual

$$\mathbf{F}(\mathbf{v},\mathbf{x}) = \mathrm{A}(\mathbf{v})\mathbf{x} = \mathbf{0} \tag{6.8.8}$$

It is easy to modify this residual: we have

$$\mathbf{F}_m(\mathbf{v},\mathbf{v}^*,\mathbf{x}) = \mathrm{M}(\mathbf{v},\mathbf{v}^*)\mathrm{A}(\mathbf{v})\mathbf{x} = \mathbf{0} \tag{6.8.9}$$

Solving Eq. 6.8.9 is another nonlinear multiparameter eigenvalue problem, which can then be solved with the same SLP algorithm or singular vector iteration. No modifications to the algorithms are required.

6.8.2 Numerical experiments

We apply these methods of deflation to the section model with Theodorsen aerodynamics. We will consider the SLP algorithm as our test case, as the analysis of the effect of deflation on each different algorithm would take a long time. We apply this algorithm to the section model with Theodorsen aerodynamics, restricted as per Section 6.7 with $\chi_c = \Upsilon_c = 0.1$. We will use norm deflation. Figure 6.29 shows the iteration paths for nine initial guesses around one of the nonphysical flutter points, applied to the restricted Theodorsen system with no deflation. As can be seen, all of the initial guesses lie within the basin of attraction of the nearby flutter point.

We now deflate this nearby nonphysical flutter point with norm deflation. We use a numerical estimate of this flutter point computed by one of the previously-convergent iterations, accurate to six significant figures (note that this is not equivalent to a tolerance of $\epsilon = 10^{-6}$). The resulting iteration paths from the same initial guesses are shown in Figure 6.30. None of the iterations now converge to the nearby flutter point; a good indication that the deflation is working correctly. Four initial guesses converge to the divergence point, two to the first flutter point, and three to the other nonphysical flutter point. However, it should be noted that many of the iterations show highly non-monotonic convergence, arcing out huge trails in the left half plane before returning to one of the flutter points near the χ-axis. This is a worrying trend which will continue.

Figure 6.31 then shows the iteration paths for the same initial guesses but now with both nonphysical flutter points deflated (to the same accuracy, six significant figures). Four of the iterations now converge to the divergence point, three to the first flutter point, and two do not converge. Those iteration paths that converge do so in a reasonable manner; the non-converging ones arc out large curves in the left-half plane, but do not return with the specified maximum number of iterations.

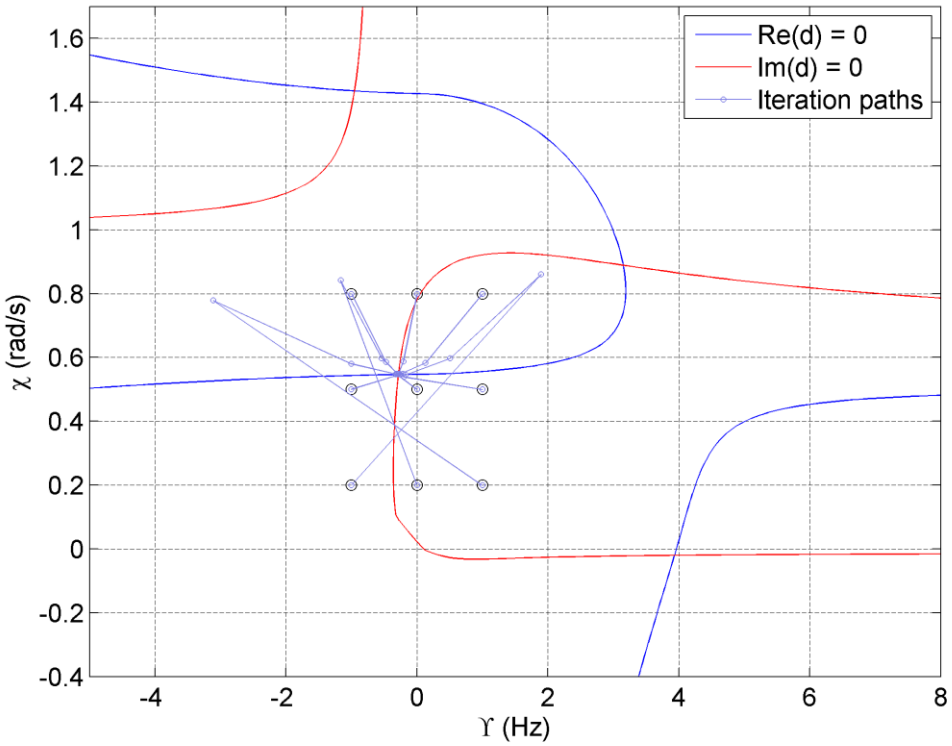


Figure 6.29: Contour plot with nine iteration paths for the SLP algorithm (Υ-χ section model, restricted with $\chi_c = 0.1$ rad/s and $\Upsilon_c = 0.1$ Hz, with no deflation).

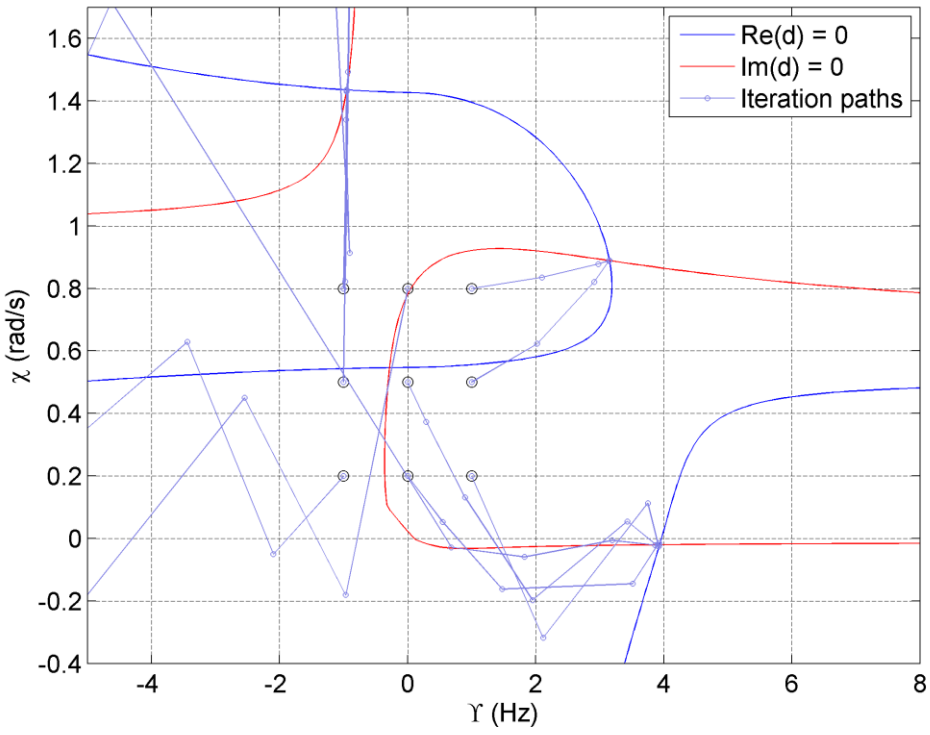


Figure 6.30: Contour plot with nine iteration paths for the SLP algorithm (Υ-χ section model, restricted with $\chi_c = 0.1$ rad/s and $\Upsilon_c = 0.1$ Hz, with one flutter point deflated).

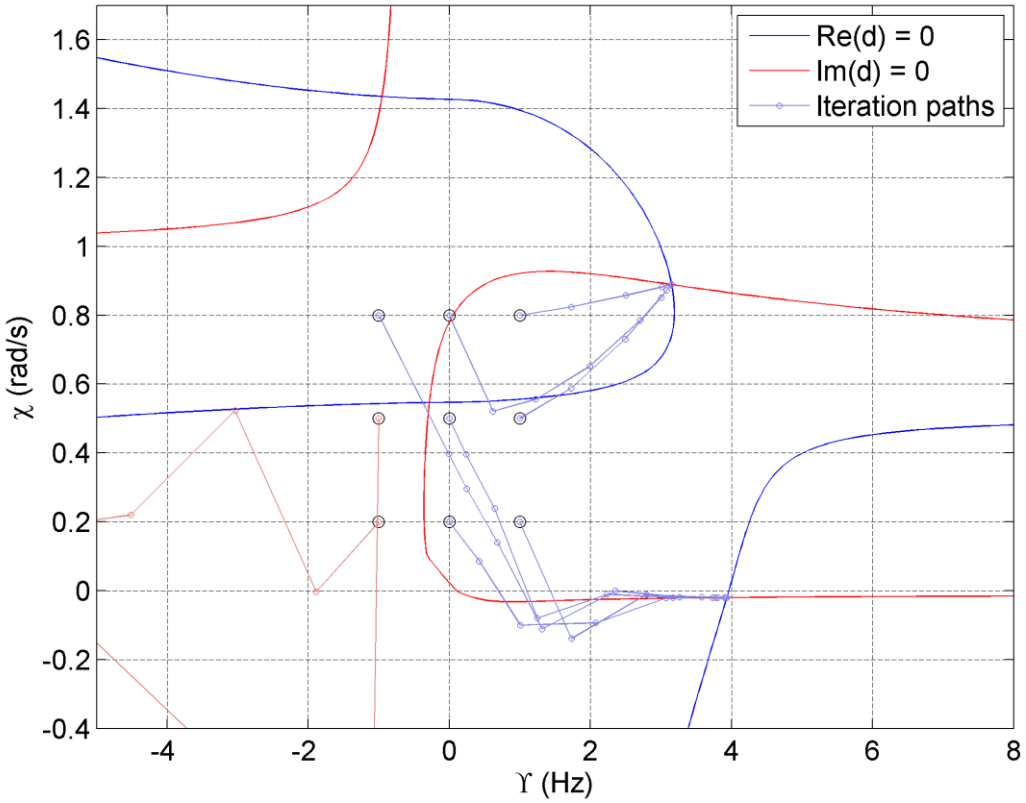


Figure 6.31: Contour plot with nine iteration paths for the SLP algorithm (Υ-χ section model, restricted with $\chi_c = 0.1$ rad/s and $\Upsilon_c = 0.1$ Hz, with two flutter points deflated).

Finally, Figure 6.32 shows the iteration paths for the same initial guesses but with both nonphysical flutter points and the divergence point deflated. Three iterations converge to the first flutter point, three to the restabilisation point at high Υ, and three do not converge. Although we have seen an improvement in the number of flutter points converging to the first flutter point (zero, two, three, three for the successive deflations) we have also increased the number of iterations that do not converge. The basin of attraction of the first flutter point is not being significantly enlarged. As a concrete test of this, Figures 6.33 – 6.37 show numerical convergence analyses for the section model with Theodorsen aerodynamics, with flutter points successively deflated until only the first flutter point is left. Note that we denote the nonphysical flutter point with the lowest χ as nonphysical point 1, and the other as nonphysical point 2. The deflation occurs in this order: nonphysical point 1, nonphysical point 2, restabilisation point, divergence point.

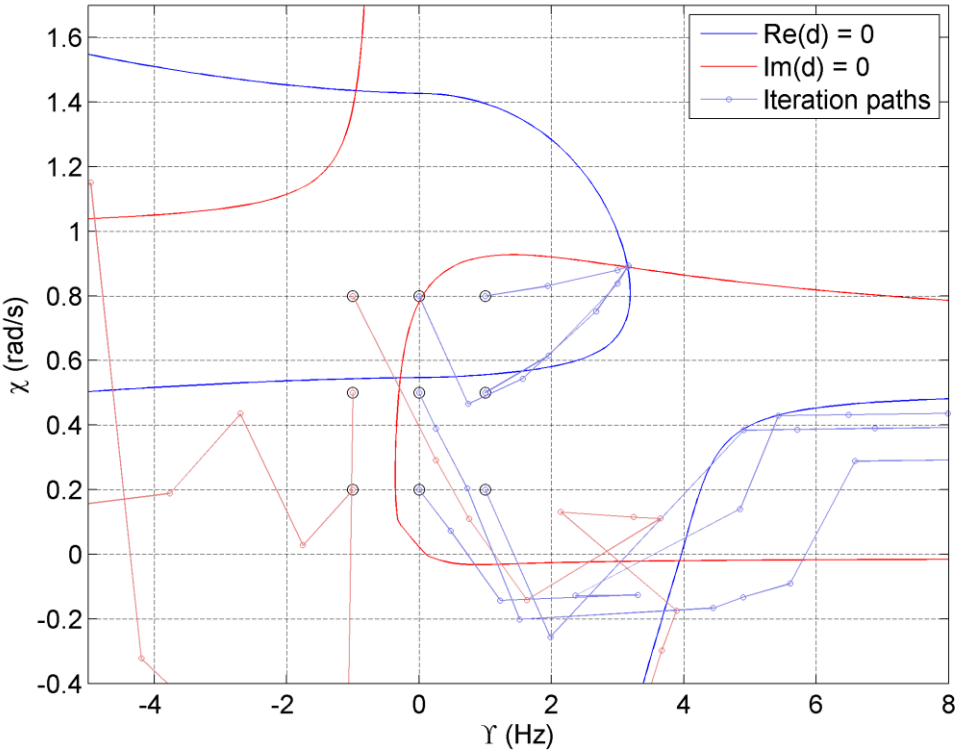


Figure 6.32: Contour plot with nine iteration paths for the SLP algorithm (Υ-χ section model, restricted with $\chi_c = 0.1$ rad/s and $\Upsilon_c = 0.1$ Hz, with three flutter points deflated).

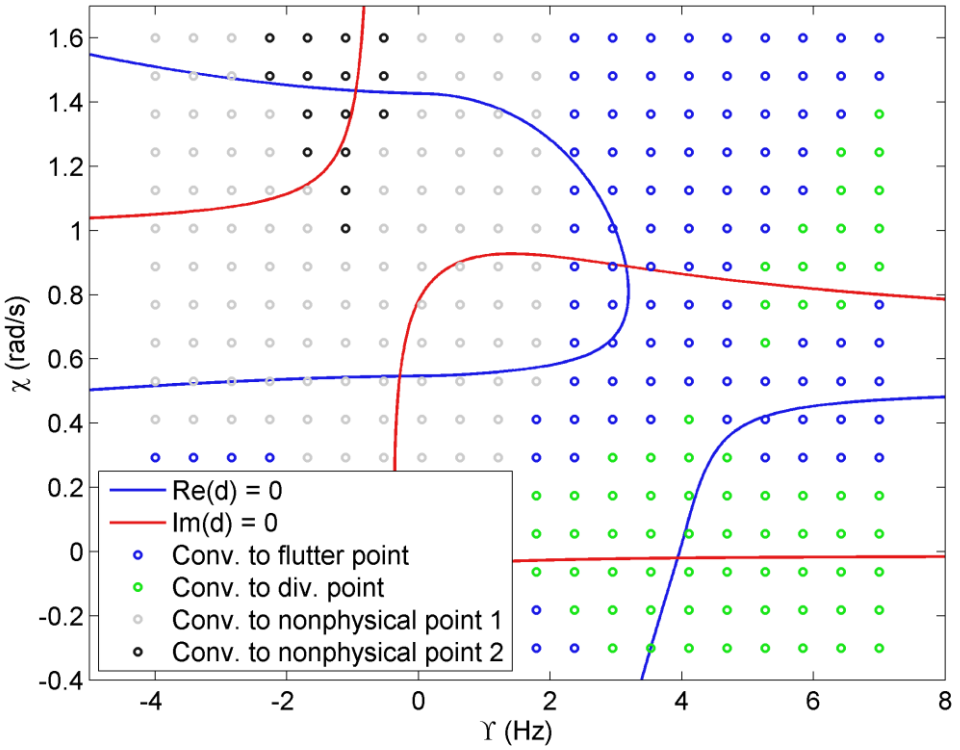


Figure 6.33: Numerical convergence plot for the section model with Theodorsen aerodynamics, restricted, with no deflation.

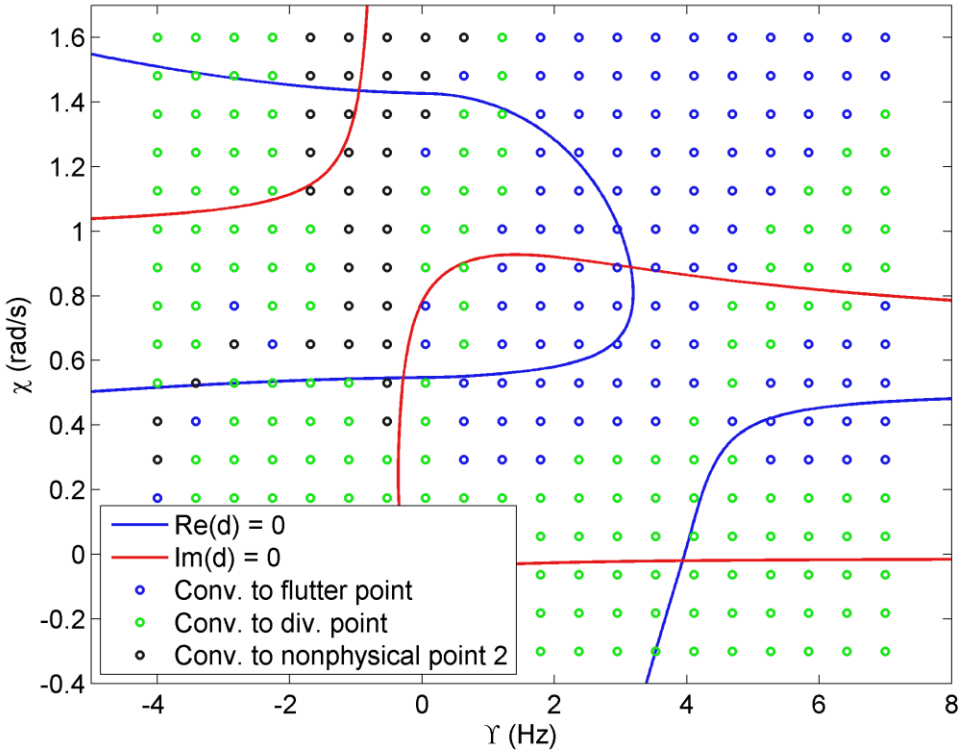


Figure 6.34: Numerical convergence plot for the section model with Theodorsen aerodynamics, restricted, with nonphysical point 1 deflated.

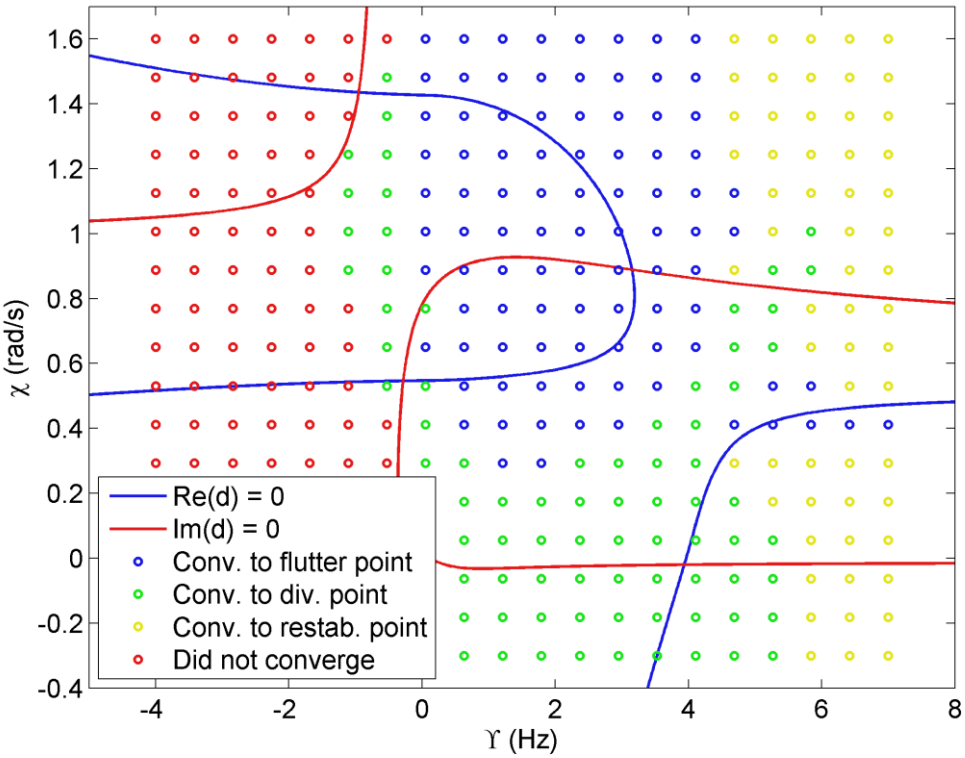


Figure 6.35: Numerical convergence plot with nonphysical points 1 and 2 deflated.

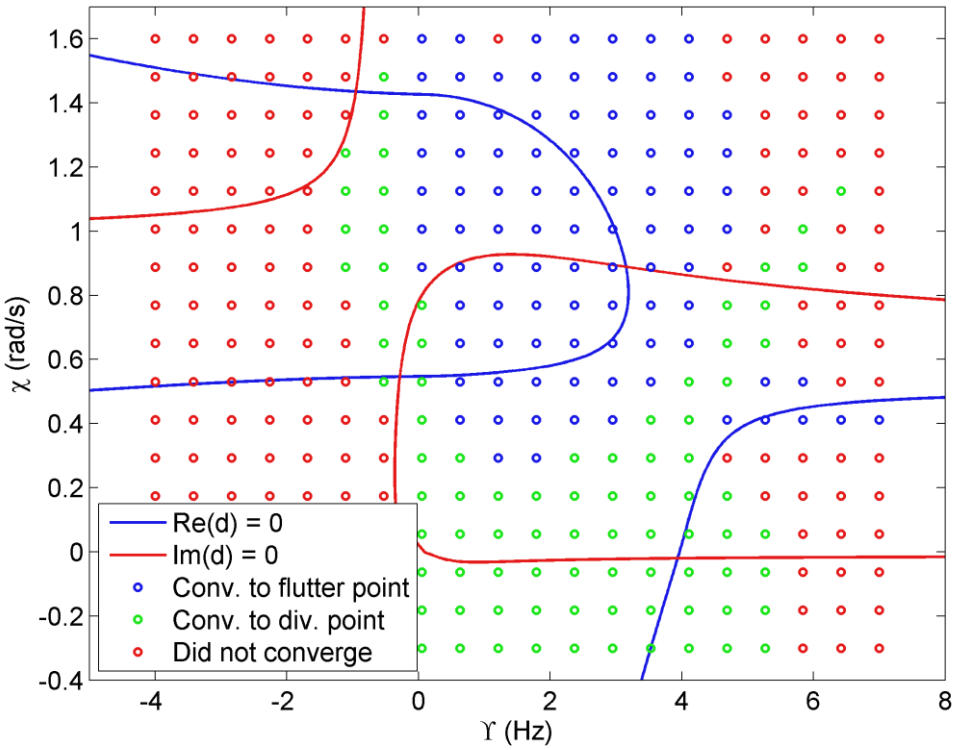


Figure 6.36: Numerical convergence plot with the restabilisation point and nonphysical points 1 and 2 deflated.

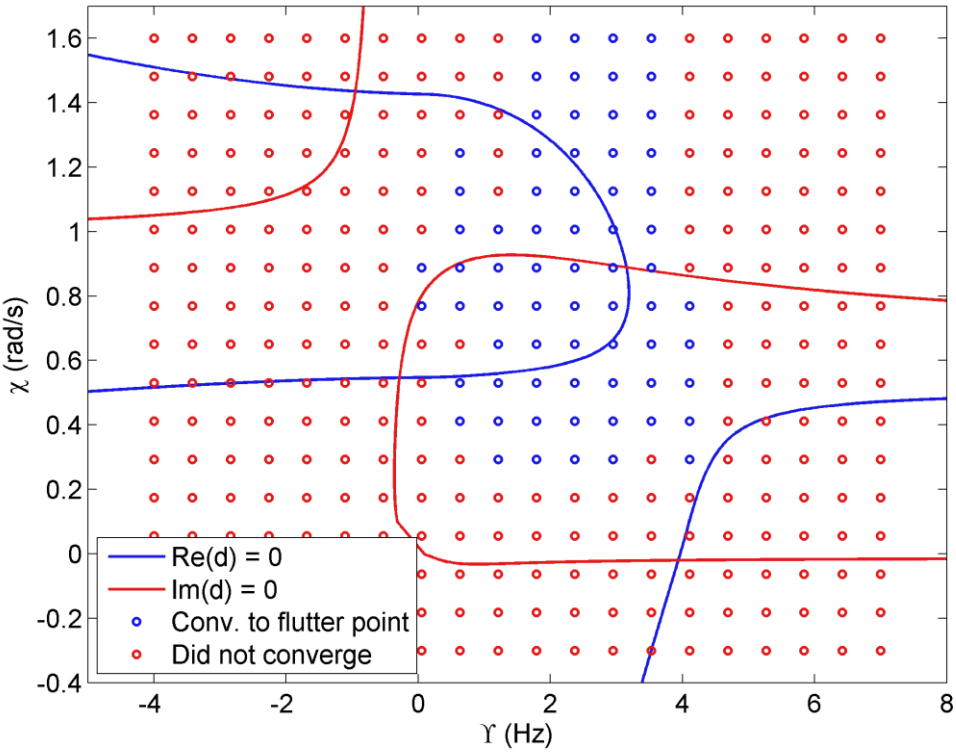


Figure 6.37: Numerical convergence plot with the restabilisation point and nonphysical points 1 and 2 deflated.

As can be seen, the successive deflations do succeed in preventing the iterations from converging to the deflated flutter points. They are less successful at broadening the basins of attraction of other flutter points. We note that the basin of attraction for the first flutter point in Figure 6.33 is actually larger than that in Figure 6.37. The first deflation event (Figure 6.34) is successful – it removes nonphysical flutter point 1 and enlarges the basins of attraction of the other points – but the deflation events after that introduce larger and larger areas of non-convergence. Given that this deflation method works by introducing poles in the residual near the flutter point, it may be that the system's residual function is simply getting too crowded, and the residual poles are obscuring the remaining flutter points. If this is so, then it is vital that the number of extraneous flutter points in the system be kept to a minimum – this provides another motivation for the development of our restriction algorithm in Section 6.7. It may be possible to devise more robust deflation methods: some promising work for the one-parameter problem has been done by Effenberger [19] and it may be possible to extend this to the multiparameter case. This is a very interesting area for future research.

6.9 ASSESSMENT AND CONCLUSION

6.9.1 General assessment

Apart from Chapter 1, when we introduced the relationship between multiparameter spectral theory and aeroelasticity, this is probably the most novel chapter in this thesis. None of the algorithms presented in this chapter have been applied before to unstructured multiparameter problems, and only one (the iterated contour plot) has been presented before for linear multiparameter systems. This chapter should thus have relevance not only for aeroelasticians, but for mathematicians working in multiparameter spectral theory.

Making recommendations as to which of these methods would be superior for solving particular aeroelastic problems is not easy: we have not been able to devise an 'ideal' solver that is clearly better than all others. Rather, we have a series of different methods with particular advantages and disadvantages. Moreover, we have not considered enough different types of systems to give a truly general picture. Nevertheless, at the very least we can compare the relative merits of these solvers.

The successive linear problems (SLP) algorithm, our first algorithm, generally performed well. When applied to the τ-λ form of the Theodorsen system, the convergence basin to the first flutter point occupies almost the entire upper-right quarter plane. This is very positive. However, when applied to the Υ-χ form, which has a divergence point to distract these upper-right quarter plane iterations, the convergence basins of the first flutter point and the divergence point mix chaotically. Though the combined convergence basin is large, this chaotic behaviour is not good as it means that there are no locations where the iterations can be reliably expected to converge to one point over the other. However, the convergence rate is generally good: the method shows second-order convergence to the flutter points, and first-order convergence to the divergence point (as a result of the system's nondifferentiability at $\chi = 0$). Also, the computational cost of a single iteration of the SLP algorithm is higher than that of the other algorithms we have presented, as one has to solve a linear multiparameter system at each step.

The iterated contour plot shows very similar convergence rate properties to the SLP algorithm, but instead of a chaotic first flutter point / divergence point basin of attraction, it just shows a smaller (but more reliable) basin of attraction to the first flutter point. However, note that a similar chaotic phenomenon occurs between the basins of convergence to the nonphysical flutter points and the basins of non-convergence in the left diagonal half of the upper-right quarter plane. While for this system we are not interested in these convergence basins, in other systems this chaotic behaviour may occur in basins of attraction which we are interested in. The analysis of this chaotic behaviour is an interesting area for future theoretical research. Note also that the convergence basin to the divergence point is significantly less dominant in the iterated contour plot algorithm than in the SLP algorithm. However, the combined basin of attraction for the nonphysical flutter points is much more dominant in the iterated contour plot. If it is the first flutter point that is desired, then this simply becomes a choice of evils. If you chose the SLP algorithm, then it may be slightly more difficult to find the first-flutter point basin, but at least if you don't then you will probably still get some useful physical information (the location of the divergence point or second flutter point). If you chose the iterated contour plot, then although it may be easier to find the first-flutter point, all those iterations that do not find it

will either not converge or will converge to flutter points which are of no interest. These choices are also related to the availability (or lack) of a good deflation algorithm.

One major problem – or at least, potentially problem feature – of this algorithm is its explicit computation of the determinant. On the one hand, the use of the determinant means that the computational cost per iteration is low – only a 2×2 linear system has to be solved at each step, irrespective of the size of the system. The computational cost of determinant evaluation is not negligible – for a standard evaluation (based on LU decomposition) the computational complexity is $\mathcal{O}(n^3)$ [11] – but certainly not as costly as the $\mathcal{O}(n^6)$ needed in the SLP algorithm. However, it may also restrict the size of the system that can be solved – there may be problems with the accuracy of the determinant evaluation when the system is large. It has often been noted in mathematical literature that the determinant "gives no indication as to the singularity of the system". This point is indicated usually by considering matrices of radically different forms, scaling or sizes. However, it is not clear that such a criticism is (in itself) valid in the cause of the iterated contour plot algorithm. We are not using the determinant as a convergence criterion, to *measure* singularity – we use the inverse condition number for that. We only rely on the fact that, for a given matrix function, minimising the determinant of that function is likely to bring that system closer to singularity. If the function is pathological then the algorithm is likely to fail (as we would expect), but for aeroelastic systems this is very improbable. The problem with the determinant is that it will lose accuracy due to the effects of finite-precision arithmetic (if the system is large or consists naturally of elements with very small value, then the determinant evaluation). Normalising each row of the matrix during each iteration (or after m iterations) seems at least a superficially attractive solution, but this may have side effects, including an increased computational expense and the possibility that it may affect the relationship between determinant minimisation and singularity (e.g. steps that increase the condition number of the matrix may register as increasing the determinant due to rescaling). This is an interesting area for future investigation.

The second-order contour plot is essentially a failure: while there is a well-defined basin of attraction to the first flutter point, this basin is small. It maintains the second-order convergence to flutter points and first-order convergence to divergence points of the first-

order method as does not noticeably improve. It is not recommended for any system, though it may prove useful as a basis for further research on higher-order algorithms.

The singular vector iteration (SVI) shows the strongest attraction to the first flutter point of all the methods. The basin of attraction occupies the majority of the upper-right quarter plane even when a divergence point is present in the system. However, this comes at the cost of slow first-order convergence. This slow convergence is somewhat offset by the relatively low cost of each iteration – the singular value decomposition for a square matrix has complexity $\mathcal{O}(n^3)$ [11], and this is probably the most intensive aspect of the algorithm. It may also be possible to speed up convergence using either over-relaxation or Aitken's Δ^2 acceleration; both of which are known to increase the convergence rate of first-order methods. However, as it stands, the SVI algorithm is the most robust of the algorithms we have developed so far, in terms of the reliability of its convergence to the first flutter point. Table 6.1 presents a general overview of our assessment so far of these unstructured algorithms

Table 6.1: Overview of algorithm features

Successive linear problems	• Strong attraction to first flutter point τ-λ form, erratic attraction in Υ-χ form. • Second-order convergence to flutter points, first-order convergence to divergence points. • High cost per iteration.
Iterated contour plot	• Moderate attraction to first flutter point • Basins of attraction to nonphysical flutter points still show erratic convergence. • Second-order convergence to flutter points, first-order convergence to divergence points. • Low cost per iteration.
Iterated contour plot (second-order)	• Weak attraction to first flutter point. • Large areas of nonconvergence. • Second-order convergence to flutter points, first-order convergence to divergence points. • Not recommended for practical problems.
Singular vector iteration	• Strongest attraction to first flutter point of all the methods. • Slow first-order convergence to flutter and divergence points. • Low cost per iteration.

6.9.2 Conclusions

In this chapter we have presented several solution methods for unstructured multiparameter eigenvalue problems. All of these methods are novel in that they have not been applied to unstructured multiparameter problems before, though some have been previously applied to linear multiparameter problems or nonlinear one-parameter problems. The singular vector iteration – one of our most successful methods – is not directly based on any existing one-parameter or multiparameter algorithm, though it shares some similarities with the tensor Rayleigh quotient iteration. In this chapter we also devised a deflation method for our algorithms, and we have introduced the concept of *restriction* – something that is not based on any existing techniques. However, apart from simply devising these algorithms, we have also tested them thoroughly on the section model with Theodorsen aerodynamics. Using this model we were able to make numerical measurements of convergence properties such as convergence order, and to observe more qualitative properties such as the size of the convergence basins. The deflation and restriction algorithms were also tested on these models. Finally, in Section 6.9.1 we presented an assessment of the properties of the solution methods that we have devised. We will discuss these methods further and make recommendations as to their use in Chapter 7.

6.10 REFERENCES

[1] Hochstenbach, M. E., Kosir, T., and Plestenjak, B., 2004, “A Jacobi-Davidson Type Method for the Two-Parameter Eigenvalue Problem,” SIAM Journal on Matrix Analysis and Applications, **26**(2), pp. 477–497.

[2] Meerbergen, K., and Plestenjak, B., 2014, A Sylvester-Arnoldi type method for the generalized eigenvalue problem with two-by-two operator determinants, Department of Computer Science, KU Leuven.

[3] Podlevskii, B. M., 2008, “Newton’s method as a tool for finding the eigenvalues of certain two-parameter (multiparameter) spectral problems,” Computational Mathematics and Mathematical Physics, **48**(12), pp. 2140–2145.

[4] Plestenjak, B., 2000, “A Continuation Method for a Right Definite Two-Parameter Eigenvalue Problem,” SIAM Journal on Matrix Analysis and Applications, **21**(4), pp. 1163–1184.

[5] Ruhe, A., 1973, “Algorithms for the Nonlinear Eigenvalue Problem,” SIAM Journal on Numerical Analysis, **10**(4), pp. 674–689.

[6] Adams, R. A., and Essex, C., 1991, Calculus of several variables, Addison-Wesley, Don Mills, Ontario, Canada.

[7] Quarteroni, A., Sacco, R., and Saleri, F., 2007, Numerical mathematics, Springer-Verlag Berlin, Berlin, Germany.

[8] Podlevskyi, B. M., 2010, “Numerical solution of some two-parameter eigenvalue problems,” Journal of Mathematical Sciences, **165**(2), pp. 214–220.

[9] Dai, H., 1999, “Numerical methods for solving multiparameter eigenvalue problems,” International Journal of Computer Mathematics, **72**(3), pp. 331–347.

[10] Khazanov, V. B., 2005, “Methods for solving spectral problems for multiparameter matrix pencils,” Journal of Mathematical Sciences, **127**(3), pp. 2033–2050.

[11] Golub, G. H., and Van Loan, C. F., 2013, Matrix computations, The Johns Hopkins University Press, Baltimore, Maryland, USA.

[12] Sewell, G., 2005, Computational methods of linear algebra, Wiley-Interscience, Hoboken, New Jersey, USA.

[13] Petersen, K. B., and Pedersen, M. S., 2012, The Matrix Cookbook, Technical University of Denmark, Kongens Lyngby, Denmark.

[14] Peitgen, H.-O., Saupe, D., and Barnsley, M. F., eds., 1988, The Science of fractal images, Springer-Verlag New York, New York, New York State, USA.

[15] Bernstein, D. S., 2009, Matrix mathematics theory, facts, and formulas, Princeton University Press, Princeton, New Jersey, USA.

[16] Brown, K. M., and Gearhart, W. B., 1971, “Deflation techniques for the calculation of further solutions of a nonlinear system,” Numerische Mathematik, **16**(4), pp. 334–342.

[17] Farrell, P. E., Birkisson, A., and Funke, S. W., 2014, Deflation techniques for finding distinct solutions of nonlinear partial differential equations, University of Oxford, Oxford, UK.

[18] Anton, H., 1977, Elementary Linear Algebra, John Wiley & Sons, New York, New York State, USA.

[19] Effenberger, C., 2013, “Robust solution methods for nonlinear eigenvalue problems,” Doctoral Dissertation, École polytechnique fédérale de Lausanne.

Chapter 7
Concluding remarks

7.1 INTRODUCTION

In this chapter we summarise and discuss our findings. We also discuss some interesting issues that have been raised along the way – particularly, the problem of constrainedness. We discuss further extensions to our central thesis (Chapter 1, Section 1.2, pp. 6-8), and avenues for future research. Page numbers are included in section and chapter references, for ease of navigation.

7.2. ASSESSMENT OF METHODS

In this thesis we presented eight different methods for solution of flutter problems: one for the solution of linear flutter problems, three for the solution of semi-structured problems[1], and four for the solution of unstructured problems. We also presented two different types of polynomial linearisation, and we discussed a variety of other methods of solution (derived usually from one-parameter spectral theory) that could be easily generalised to the multiparameter case. For the practicing aeroelastician, the question naturally arises of which of these methods to use, given a particular flutter problem. We address this question in this section, and we also identify strengths and deficiencies in the methods that we have presented, as well as possible avenues for their improvement.

7.2.1 Solvers for structured problems

The operator determinant method is a robust and effective way of computing the flutter points of structured systems. However, it can be computationally expensive, especially for larger systems. When applied to nonsingular systems, the method has computational complexity $\mathcal{O}(n^6)$. Moreover, we noted in Chapter 4 (pp. 89-90, 92) that the application of the method to singular systems (via the compression algorithm, Chapter 4, Section 4.2.4, pp. 62-63) massively increases the computation time. This does not appear to have been noted

[1] Including the method based on approximation of the semi-structured problem by accurate rational or fractional-order expressions (Chapter 2, Section 2.3.5, pp. 23ff. and Chapter 4, Section 4.4, pp. 80ff.).

by the existing literature on this subject. However, as we also noted in Chapter 4 (Section 4.5, pp. 93-94), several other methods for the solution of linear multiparameter eigenvalue problems have been proposed. Many of these are based around optimising the solver for the operator determinant generalised eigenvalue problem (GEP), given that the structure of these operator determinants is known. These methods have significantly lower computational complexity, down below $\mathcal{O}(n^3)$ [1]. Several of these methods have been implemented in a MATAB toolbox for structured multiparameter eigenvalues, *MultiParEig* [2], and are sufficiently robust that they may be applied. However, note that all of these methods have so far been developed only for two-parameter problems. Some of these methods show potential for generalisation to n-parameter problems (the subspace methods), however these generalised methods have not yet been devised. Other methods, such as those of Sylvester-Arnoldi type, show less potential for generalisation as they rely on being able to reformulate the operator GEP into a different form of matrix equation (for which other well-known solvers are available). As the operator determinant definition becomes more complex, the resulting matrix equation changes and efficient solvers may not be available.

Several of these methods also have problems dealing with singular systems. The Sylvester-Arnoldi algorithms presented in [1] will fail when the determinant Δ_2 is singular. Note that singularity or otherwise of a nonhomogeneous multiparameter problem is governed not by Δ_2 but by Δ_0, and so the condition that Δ_2 must be nonsingular does not equate to the condition that the problem must be nonsingular (nor vice versa). Ref. [1] presents a technique, based on introducing a shift parameter into the equations, to solve problems with singular A_i but nonsingular Δ_2. However the restriction to nonsingular Δ_2 is a crippling disadvantage, as almost all of our linearised polynomial systems will be singular (see Chapter 4, Section 4.3, pp. 67ff.). Given that the algorithms operate on the coefficient matrices themselves (and not directly on the operator determinants) it is not easy to see how the compression algorithm presented in Chapter 4 (Section 4.2.4, pp. 62-63) could be applied. Thus, while the Sylvester-Arnoldi algorithms could be useful as part of an unstructured or semi-structured solver (where the problem solved at each step is seldom singular) they will usually be unavailable for the solution of linearised polynomial problems – which is precisely where they are needed, as these linearisations can become very large.

The subspace methods are little better in this regard. The Jacobi-Davidson method presented in [3] applies only to nonsingular problems, though this is already an improvement over an earlier version of this method [4] which required the system to be right-definite (a condition defined in [4]) which implies nonsingularity but not vice-versa. The Rayleigh-Ritz method presented in [5] would not appear to be invalid for singular problems, as it relies on the definition of the Rayleigh quotient with the operator determinants defined via their Kronecker product definition – though we note that Ref. [5] makes no note of the method's applicability or lack thereof to singular problems. Again, there would appear to be no possibility of applying the compression algorithm to either of these methods, as they both operate on the system matrices and not the operator determinants. If fact, we would generally be able to say that, in the literature as it stands, the solution of singular problems is incompatible with the desire to make the solution algorithm more efficient. This is for the following reason. The only way of dealing with singular problems that is currently known is the compression algorithm of Muhič and Plestenjak [6], which applies directly to the operator determinants. That is, from the original operator determinant pencils of the singular problem, $\Delta_{1\mathrm{s}} - \lambda\Delta_{0\mathrm{s}}$ and $\Delta_{2\mathrm{s}} - \lambda\Delta_{0\mathrm{s}}$, we extract two new pencils, $\Delta_1 - \lambda\Delta_0$ and $\Delta_2 - \lambda\Delta_0$, representing the finite regular part of the problem. If we wish to make the GEP solver more efficient, we need to be able to break down the operator determinants into their matrix-coefficient components ($\Delta_{0\mathrm{s}} = \mathrm{B}_1 \otimes \mathrm{C}_2 - \mathrm{C}_1 \otimes \mathrm{B}_2$, etc.) and observe their structure. However, by passing the pencils through the compression algorithm we have obscured their structure significantly, and the usual relationships such as $\Delta_0 = \mathrm{B}_1 \otimes \mathrm{C}_2 - \mathrm{C}_1 \otimes \mathrm{B}_2$ do not apply. Thus it is not possible to make the solver more efficient, as nothing further can be said about the matrix structure.

There are several possible approaches to this problem. The first, which we have already alluded to, is to forget (for the moment) about making the GEP solver more efficient, and to concentrate on making the compression algorithm more efficient. As we have noted earlier and in Chapter 4 (Section 4.2.4, pp. 89-90, 92), it is the compression algorithm which occupies the vast majority of the computational effort required to solve a large singular system. A significant portion of the body of literature on solvers for linear multiparameter systems – a very small body of literature, admittedly – has focused on devising more

efficient GEP solvers. However this is not actually the problem that is currently relevant to the application of these solvers to aeroelastic or indeed other physical systems. The most relevant problem at this point in time is the high computational cost of the compression algorithm, and it is this problem to which it would be wiser to devote research effort. A related approach which we might also be interested in is to attempt to derive new compression algorithms that act not on the operator determinants but on the system matrices themselves – for example, can we devise an algorithm that will construct an equivalent nonsingular multiparameter *system* (not just a set of operator determinants) that represents the finite regular parts of the operator determinant pencils of a singular system? If we could, then this would allow us to use the efficient solvers for nonsingular systems that we have been discussing to solve nonsingular systems.

Lastly, an entirely different approach would be to devise solvers that do not require the condition of nonsingularity. We have, in fact, already met some of these in the form of our unstructured solvers in Chapter 6 (pp. 124ff.). The iterated contour plot methods, when applied to linear problems, does not require problem nonsingularity (i.e. nonsingularity of Δ_0). However, they do require a different nonsingularity condition, namely that the matrix function $\mathrm{A}(\chi, p, \dots)$ should not be singular for all $(\chi, p, \dots)$. These conditions are not equivalent. The singular vector iteration can easily be modified to solve singular systems: at each iteration step, one selects not the singular vector associated with minimum singular value, but simply that which is associated with the minimum *nonzero* singular vector. This should converge to the finite-regular eigenvalue of the system (cf. Chapter 4, Eq. 4.2.16, p. 62). However, the problem with using these unstructured methods is that they are less robust, and may have problems with convergence – particularly with regard to the deflation of computed eigenvalues (see Chapter 6, Section 6.8.2, pp. 175ff.). However, if we are applying the problem to a known, algebraically-defined linear system, it may be possible to devise more robust deflation methods. This is, again, an interesting area for future research effort.

In short, we would say that the most significant deficiency of existing linear multiparameter solvers is their applicability to systems that are singular in some way (whether in Δ_0 or in some other measure). Some methods cannot solve such systems; others can but only with a

high computational cost. However, we have noted, the solution of singular multiparameter eigenvalue problems has only been achieved very recently (2010 [6]), and so it is likely that the next few years will bring significant developments in this area. For practicing aeroelasticians, however, the only reliable method that is currently available for nonsingular problems is the operator determinant method, and so this method should be the first port of call. Other methods may be useful for large systems, but are only available when these systems are nonsingular.

7.2.2 Solvers for semi-structured systems

In this thesis we presented two distinct classes of semi-structured methods. The first class, presented in Chapter 2 (Section 2.3.5, p. 23ff.) and Chapter 4 (Section 4.4, p. 80ff.), approximated the unstructured element in the problem (in our case, Theodorsen's function) to a high degree of accuracy, and solved the resulting high-order polynomial system via linearisation and a direct linear solver (the operator determinant method). The advantage of this method is its directness, and consequentially the fact that all eigenvalues can be computed simultaneously and reliably. This is an extremely significant advantage, as the single most important problem facing both the other class of semi-structured solvers (the iterative solvers) and the unstructured solvers is how to reliably compute multiple eigenvalues – or indeed, even a single eigenvalue with a given property. The direct semi-structured methods remove this problem by their very definition. The directness of these methods does have an associated disadvantage, namely that the desired solution accuracy cannot be specified in the solution algorithm (as it can be with iterative algorithms) because this accuracy is dependent entirely on the accuracy of the unstructured element approximation. This is a very minor disadvantage, as the accuracy that we obtain using common approximations to Theodorsen's function is more than sufficient for industrial purposes (see Chapter 4, pp. 84-85). Of more concern is the required computation time – 20 seconds (unacceptably high) and 0.2 seconds (still disappointingly high) for the two approximations that we used. Indeed, the solution accuracy is rather too high, and we would be happy with significantly lower accuracy in exchange for reduced computation time.

To reduce computation time we could focus on developing faster direct solvers (especially for singular systems, with or without a compression algorithm). Indeed, the development of better semi-structured algorithms of this class is intertwined with the development of better linear solvers, such as we discussed in Section 7.2.1 (pp. 188ff.). However, another possibility for reducing the computation time (but at the expense of accuracy) is to use simpler approximations of Theodorsen's function. Unfortunately, the devising of low-accuracy (but very simple) approximations of Theodorsen's function is not one that has been explored previously in the literature: this is an area where a small amount of research would have significant impact. We note that our methods are able to handle a considerably broader class of approximations than are usually considered (cf. for example [7]), including approximations with fractional powers.

Of course, the other class of semi-structured methods that we have presented are the iterative solvers. These methods utilise a low-accuracy local approximation (we have used a Taylor expansion) which is then refined according to the iteration data. These methods are thus necessarily iterative, but they can have a significantly lower computation time than the direct methods, and the desired solution accuracy can be specified in the iteration tolerance. In Chapter 5 (pp. 100ff.) we presented two iterative semi-structured solvers: a Picard and a Newton method. The first-flutter point basins for these methods are excellent if the first flutter point is sufficiently isolated (e.g. in the Theodorsen τ-λ form) but poor otherwise. Hence the motivation for increasing the accuracy of the local approximation, for example by including some global components (see Chapter 5, Section 5.4, pp. 121-122). A hybrid local / global approximation of sufficient accuracy would allow the iterative method to select which flutter point to converge to – something which we greatly desire in an iterative method, but have not yet been able to obtain. This is another very interesting area for future research.

7.2.3 Solvers for unstructured problems

The crucial problem that governs the applicability of the unstructured solvers is their reliability of convergence to the desired flutter point – almost inevitably the first flutter point. This problem acquires its particular importance due to the current lack of a reliable deflation for unstructured multiparameter eigenvalues: the deflation algorithm we have

presented is useful but increasingly disrupts iterative convergence as more flutter points are deflated. If the system's flutter behaviour is relatively uncluttered (e.g. a standard two-parameter flutter problem) then we would recommend that the system be transformed in the τ-λ form, and solved using the successive linear problems (SLP) algorithm. The algorithm shows very good convergence to the first flutter point in the τ-λ form. If the system's flutter behaviour is more complex (e.g. with multiple nearby flutter points that cannot be removed) but it is still only the first flutter point that is desired, then the singular vector iteration (SVI) would be recommended. This method shows good convergence to the first flutter point even when there is a divergence point in the system to compete for the basin of attraction. The downside of this method is its slow convergence. If the system's flutter behaviour is very complex, or it is thought that there may be other flutter points obscuring the point of interest, then we have no algorithm that is currently capable of reliable solving this system for the flutter point of interest. It would be better to first use the contour plot method to visualise the instability behaviour of the system. The results from the contour plot can then be used to provide suitable initial guesses near the flutter points of interest.

In all cases the restriction algorithm is recommended to be applied; depending on the available knowledge about system it may be possible to restrict the system even more tightly than we have done (e.g. with two extra bounding lines, forming an enclosed box). If this is possible then it is advised. Table 7.1 shows an overview of our recommendations as to solver usage

Table 7.1: Algorithm recommendations

Algorithm	**Recommended for:**
SLP algorithm	Uncluttered systems with no other flutter points to distract from the first flutter point
SVI algorithm	Systems with one or two other points to distract from the first flutter point
Contour plot	Systems with multiple flutter points obscuring the first flutter point

7.3 CONSTRAINEDNESS

One area of study which we have skirted throughout this thesis is the question of what constitutes a fully-constrained problem – i.e. one which has a discrete spectrum. If a problem is underconstrained then its spectrum will be continuous. However, some systems which are at least apparently fully constrained still have a continuous spectrum – note, for example, our introductory steady model. The constrainedness of the problem would appear to be related to three factors:

- The number and distinctness of the equations in the problem.
- The field occupied by the matrix – whether real or complex.
- The singularity of the problem and its coefficient matrices.

The exact relationship between these three factors is still elusive. For example, one significant question is "to what extent can the conjugate equation ($\overline{\mathrm{A}}(\chi, p) = 0$) be considered redundant?" Some of the methods that we have devised make no use of the conjugate equation (e.g. the iterated contour plot) and would need to be modified to apply to systems with two different equations. However, when computing the flutter points using a direct solver (as we did in Chapter 4, pp. 58ff.) the conjugate equation is absolutely essential.

The question of constrainedness becomes even more interesting when we consider some special cases. For example, some multiparameter problems are of the form in which they can be reduced to single-parameter ones. In the two parameter-case, if the problem is given by

$$(\mathrm{A} - \mathrm{B}\chi - c\mathrm{B}p)\mathbf{x} = \mathbf{0}, \tag{7.3.1}$$

with scalar c, it can be written as

$$\big(\mathrm{A} - \mathrm{B}(\chi + cp)\big)\mathbf{x} = \mathbf{0}, \tag{7.3.2}$$

which is a generalised eigenproblem with eigenvalue $\chi + cp$. If c is real then the eigenvalue will necessarily be real: if c is complex then we must also consider complex eigenvalues. More interestingly, if we know (for whatever reason) that the eigenvector of the system is real, then for any two-parameter problem

$$(\mathrm{A} - \mathrm{B}\chi - \mathrm{C}p)\mathbf{x} = \mathbf{0}, \tag{7.3.3}$$

we can set the real and imaginary parts of this equation to be zero

$$\begin{aligned} (\mathrm{Re}(\mathrm{A}) - \mathrm{Re}(\mathrm{B})\,\chi - \mathrm{Re}(C)\,p)\mathbf{x} = \mathbf{0} \\ (\mathrm{Im}(\mathrm{A}) - \mathrm{Im}(\mathrm{B})\,\chi - \mathrm{Im}(C)\,p)\mathbf{x} = \mathbf{0}. \end{aligned} \tag{7.3.4}$$

If we premultiply the second equation by a matrix P such that $\mathrm{P}\,\mathrm{Im}(\mathrm{B}) = -\mathrm{Re}(\mathrm{B})$, then we can substitute it into the first equation to obtain

$$(\mathrm{Re}(\mathrm{A}) + \mathrm{P}\,\mathrm{Im}(\mathrm{A}) - (\mathrm{Re}(C) + \mathrm{P}\,\mathrm{Im}(C))p)\mathbf{x} = \mathbf{0}, \tag{7.3.5}$$

which is again a generalised eigenproblem in p, of size $n \times n$. This is a considerable improvement on the operator determinant GEP. The problem here is identifying that the eigenvector will be real (even though the matrix coefficients are complex). This will not generally be the case. However, both of these concepts indicate that the field occupied by the matrix coefficients in the problem (real or complex) has a significant effect on the constrainedness of the problem. There is much future research to be done in this area.

7.4 FURTHER EXTENSIONS

In Chapter 1 (Section 1.2, pp. 6-8) we discussed the central concept of this thesis, that being the relationship between aeroelastic flutter and multiparameter spectral theory. We also discussed a variety of extensions to the central thesis – for example, the extension to flutter problems with more than two eigenvalue parameters. With the past six chapters of experience, we now revisit this topic and look at other possible extensions to our central thesis.

One very interesting possibility involves the computation of not only flutter points, but points at any given modal damping. If we have a linear problem in $\chi \in \mathbb{C}$ and $p \in \mathbb{R}$, then we can split the complex χ into its real and imaginary parts ($\chi = \chi_R + \iota\chi_I$), in which case we obtain the problem

$$(\mathrm{A} - \mathrm{B}(\chi_R + \iota\chi_I) - \mathrm{C}p)\mathbf{x} = \mathbf{0}. \tag{7.3.6}$$

If we specify the desired χ_I then we obtain a two-parameter problem which can be solved via the usual methods. The resulting p would be the airspeed parameter values required to make the structure's modal damping equal to the specified χ_I.This could be very useful for computing flutter pseudo-spectra – i.e. not only flutter points, but points that are very near to becoming flutter points. Pseudo-spectra have been applied before in other instability-related field (e.g. hydrodynamic stability [8]) but have not previously been used in aeroelasticity.

Another very significant potential application is the flutter analysis of nonlinear systems. At this point this application remains only speculative; however, if the nonlinear system can be linearised into a linear or nonlinear multiparameter eigenvalue problem then it should not be difficult to devise an unstructured (or semi-structured) type solver which will iterate towards the limit-cycle oscillation, Hopf bifurcation, or whatever other instability or bifurcation is of interest. One previous paper has connected the multiparameter eigenvalue problem with the detection of Hopf bifurcations [9], and so this is potentially a useful place to start.

Lastly, one obvious extension which we have not yet mentioned is the extension of the concept of this thesis to other engineering disciplines. We have focused on aeroelasticity, but there is no reason why our algorithms cannot be used on other instability problems. There are a large number of other aeroelastic problems that we have not considered – for example, the aeroelasticity of turbomachines [10], aerothermoelasticity [11], and aeroservoelasticity [12,13]. There is also the field of hydroelasticity and naval architecture [14], and there are potentially other applications in the study of atomic-force microscopes [15]. Moreover, there are other areas of more theoretical interest, such as the study of hydrodynamic stability. There is a very large amount of research to be done in applying multiparameter methods to these problems.

7.5 SUMMARY OF FUTURE RESEARCH OPPORTUNITIES

Throughout this thesis we have been indicating avenues for future research into the links between multiparameter eigenvalue problems and aeroelastic flutter problems. We summarise the three key areas for future research here:

- Higher-order approximations for iterative methods. This includes the use of higher-order approximations in the SLP and SVI iterations, as well as the development of hybrid local / global approximations for semi-structured methods. The development of simple low-accuracy global approximations to Theodorsen's function is also of interest.
- Proofs of convergence and other theoretical results. Also of great interest would be a theoretical study into the constrainedness of flutter problems.
- Application of multiparameter solution techniques to nonlinear problems. This is a very open area for future research work.

One other less pressing but interesting areas of research is the exploration of a possible matrix-system Gaussian elimination method for flutter problems of high dimension.

7.6 FINAL CONCLUSION

In Chapter 1 (Section 1.2, pp. 6-8) we introduced the link between multiparameter spectral theory and aeroelastic flutter. This in itself is a significant novel intellectual contribution. However, in the later chapters we have expanded significantly on this initial concept, using it to devise visualisation methods and a wide variety of solvers for flutter problems. We have also assessed these solvers, applying them to real-life aeroelastic systems and measuring their performance. We then discussed and devised methods for improving these solvers.

For the iterative solvers, we devised the restriction and deflation methods to aid solver convergence. We also look at the use of higher-order local approximations to widen the convergence basins and open the possibility of selecting the desired flutter point near the start of the iterative process. For the direct solvers, we discussed methods of circumventing the large computational cost associated with the compression algorithm. All our conclusions

were supported by a variety of experimental evidence, including numerical convergence analyses, logarithmic convergence plots, and wall-clock time measurements. Finally, we assessed all of our methods, providing recommendations as to their use and future development. The large number of methods we have presented may be bewildering, so we present here a short summary of the methods we have implemented, and those we have only discussed:

Structured solvers	Implemented	
		Operator determinant method
	Discussed	
		Sylvester-Arnoldi and Subspace methods
Semi-structured solvers	Implemented	
		Picard iteration
		Newton iteration
		Approximate direct solvers
	Discussed	
		Higher-order iterative methods
Unstructured solvers	Implemented	
		Successive linear problems
		Iterated contour plot
		Second-order iterated contour plot
		Singular vector iteration
		Restriction method
		Deflation method
	Discussed	
		Various Newton-type methods

In this thesis we did several things that have never been done before. Firstly, we solved a non-trivial flutter problem with a direct solver (Chapter 4, Section 4.3.6, pp. 75ff.). The only direct solvers that have previously been presented are those that arise from classical flutter analysis, which applies only to very simple systems. Secondly, and as an extension of this first point, we solved a system with Theodorsen aerodynamics (approximated by a highly accurately) with a direct solver. This was achieved in an industrially competitive time (0.2s). The direct solution of non-trivial flutter problems has never been achieved before. Thirdly, we solved an unstructured multiparameter eigenvalue problem. Unstructured problems have not been considered before, even in theoretical literature. This result is thus of significance both for multiparameter spectral theory and aeroelasticity. However, the single most important contribution of this thesis is the opening of a whole new field of study which

stretches beyond aeroelasticity and into other industries: the treatment of instability problems using multiparameter methods. This field of research is wide and untrodden, and has the potential to change the way we analyse instability across many industries.

7.7 REFERENCES

[1] Meerbergen, K., and Plestenjak, B., 2014, A Sylvester-Arnoldi type method for the generalized eigenvalue problem with two-by-two operator determinants, Department of Computer Science, KU Leuven.

[2] Plestenjak, B., and Muhič, A., 2014, MultiParEig, University of Ljubljana, Ljubljana, Slovenia.

[3] Hochstenbach, M. E., Kosir, T., and Plestenjak, B., 2004, "A Jacobi-Davidson Type Method for the Two-Parameter Eigenvalue Problem," SIAM Journal on Matrix Analysis and Applications, **26**(2), pp. 477–497.

[4] Hochstenbach, M. E., and Plestenjak, B., 2002, "A Jacobi-Davidson Type Method for a Right Definite Two-Parameter Eigenvalue Problem," SIAM Journal on Matrix Analysis and Applications, **24**(2), pp. 392–410.

[5] Hochstenbach, M. E., and Plestenjak, B., 2008, "Harmonic Rayleigh–Ritz extraction for the multiparameter eigenvalue problem," Electronic Transactions on Numerical Analysis, **29**, pp. 81–96.

[6] Muhič, A., and Plestenjak, B., 2010, "On the quadratic two-parameter eigenvalue problem and its linearization," Linear Algebra and its Applications, **432**(10), pp. 2529–2542.

[7] Brunton, S. L., and Rowley, C. W., 2013, "Empirical state-space representations for Theodorsen's lift model," Journal of Fluids and Structures, **38**, pp. 174–186.

[8] Trefethen, L. N., Trefethen, A. E., Reddy, S. C., and Driscoll, T. A., 1993, "Hydrodynamic Stability Without Eigenvalues," Science, **261**(5121), pp. 578–584.

[9] Meerbergen, K., and Spence, A., 2010, "Inverse Iteration for Purely Imaginary Eigenvalues with Application to the Detection of Hopf Bifurcations in Large-Scale Problems," SIAM Journal on Matrix Analysis and Applications, **31**(4), pp. 1982–1999.

[10] Clark, R., Cox, D., Curtiss, H. C., Edwards, J. W., Hall, K. C., Peters, D. A., Scanlan, R., Simiu, E., Sisto, F., and Strganac, T. W., 2005, "Aeroelasticity in Turbomachines," A

Modern Course in Aeroelasticity, Kluwer Academic Publishers, Dordrecht, The Netherlands, pp. 453–490.

[11] McNamara, J., and Friedmann, P., 2007, “Aeroelastic and Aerothermoelastic Analysis of Hypersonic Vehicles: Current Status and Future Trends,” American Institute of Aeronautics and Astronautics.

[12] Karpel, M., 1999, “Reduced-Order Models for Integrated Aeroservoelastic Optimization,” Journal of Aircraft, **36**(1), pp. 146–155.

[13] Lind, R., and Brenner, M., 1999, Robust aeroservoelastic stability analysis: Flight test applications, Springer-Verlag London, London, UK.

[14] Bishop, R. E. D., and Price, W. G., 1979, Hydroelasticity of ships, Cambridge University Press, Cambridge, UK.

[15] Gutschmidt, S., and Gottlieb, O., 2010, “Bifurcations and loss of orbital stability in nonlinear viscoelastic beam arrays subject to parametric actuation,” Journal of Sound and Vibration, **329**(18), pp. 3835–3855.